

N° d'Enregistrement
au C. N. R. S.
A. O. 5256

THÈSE DE DOCTEUR INGÉNIEUR

présentée

A LA FACULTÉ DES SCIENCES

de PARIS

par

Monsieur Michel GROSSETETE

A. M. - E. S. E.

pour obtenir

le Grade de Docteur Ingénieur

Sujet de la Thèse :

CONTRIBUTION A LA SYNTHÈSE DES MACHINES SEQUENTIELLES
APPLIQUÉE A LA COMMANDE NUMÉRIQUE DES MACHINES-OUTILS

devant la Commission d'Examen

MM. ROUBINE

Président

BONNEFILLE

Examineurs

PRUDHOMME

THELLIEZ

Soutenue le 12 Février 1971

N d'enregistrement

au C.N.R.S.

A.O. 5256

THESE DE DOCTEUR INGENIEUR

présentée

A LA FACULTE DES SCIENCES

de PARIS

par

Monsieur Michel GROSSETETE

A.M. E.S.E.

pour obtenir

le grade de Docteur Ingénieur

Sujet de la Thèse : CONTRIBUTION A LA SYNTHÈSE DES MACHINES SEQUENTIELLES
APPLIQUEE A LA COMMANDE NUMERIQUE DES MACHINES-OUTILS

Soutenue le : 12 Février 1971

devant la commission d'Examen

MM. ROUBINE
BONNEFILLE
PRUDHOMME
THELLIEZ

Président
Examineurs

1840
1841

THE UNIVERSITY OF LEIDEN

LIBRARY

1840 1841

1840 1841

1840

1840 1841

1840 1841

1840 1841

1840 1841



1840 1841

1840 1841

1840 1841

1840 1841

1840 1841

1840 1841

1840 1841

A Claire,
Claudie
Pierre.



Digitized by the Internet Archive
in 2026

https://archive.org/details/IA41556004_0013

SOMMAIRE

Nous tenons à exprimer notre respectueuse gratitude à Monsieur le Professeur ROUBINE d'avoir bien voulu présider le Jury de cette Thèse.

Nous exprimons nos remerciements déferents à Monsieur le Professeur BONNEFILLE de nous avoir accepté sous sa direction.

Que Monsieur le Professeur PRUDHOMME trouve ici l'expression de notre profonde reconnaissance pour ses conseils et ses encouragements.

Soit aussi remercié Monsieur le Maître de Conférence THELLIEZ d'avoir bien voulu faire partie du Jury de soutenance de cette thèse.

Nous avons mené à bien ces travaux de recherche et la réalisation de la machine dans le cadre du Laboratoire d'Automatisme Industriel du Conservatoire National des Arts et Métiers, avec le soutien matériel de l'Ecole Nationale Supérieure d'Arts et Métiers.

Nous rendons hommage à l'équipe avec laquelle nous avons travaillé, que Messieurs KOSTKA, JAUME et GAUDIERE soient remerciés de leur participation active à l'étude et à la réalisation de la machine-outil.

Nos remerciements s'adressent à Mademoiselle CAMBERLIN pour tout le soin pris à la frappe de ce document.

SOMMAIRE

	<u>pages</u>
1 - INTRODUCTION	1
2 - MACHINES LOGIQUES DETERMINEES	7
2-1- Généralités relatives aux machines logiques	9
2-2- Diagramme de fluence des états internes notions de pannes et d'initialisation	12
2-3- Etude de machines totalement déterminées	26
2-4- Application à l'étude de machines simples	34
3 - GENERALITES SUR LA COMMANDE NUMERIQUE ETUDIEE	51
3-1- Les familles types de machines-outils	52
3-2- Structure globale de la commande numérique	53
3-3- Les ordres manuels	55
3-4- Notions de gamme d'usinage et de programmation	58
4 - SYNTAXIQUE D'UN LANGAGE MACHINE-OUTIL	61
4-1- Méthode dite de Naur et de Backus	63
4-2- L'alphabet	65
4-3- Les mots	67
4-4- Les blocs	69
4-5- Les programmes	71
4-6- Les bandes programmes codées	75
4-7- Conclusions	76
5 - ORDINOGRAMMES DE L'UNITE D'ENTREE	78
5-1- Introduction aux ordinogrammes	80
5-2- Ordinogrammes globaux	89
5-3- Développement de la phase de lecture d'un bloc.	92
5-4- Ordinogrammes de l'unité d'entrée à réaliser	104

6 - UNITES FONCTIONNELLES (Recherche des boucles essentielles)	109
6-1- Introduction à la synthèse générale	111
6-2- Introduction à la programmation manuelle et à la lecture de un caractère	116
6-3- Conclusions	122
7 - UNITE FONCTIONNELLE de Lecture de un caractère	129
7-1- Graphe aux phases d'évolutions	131
7-2- Les sous-unités classiques	133
7-3- La syntaxe de lecture	136
7-4- Unité logique de synthèse	143
7-5- Conclusions	154
8 - UNITES DE PROGRAMMATION	155
8-1- L'unité de programmation manuelle des états globaux.	157
8-2- L'unité de programmation manuelle du mode de lecture	168
9 - CONCLUSIONS	170

ANNEXE TECHNOLOGIQUE

	pages
A-1- Introduction	A-1
A-2- Circuits logiques à niveau utilisés	A-2
A-3- Types de "parasites" en fonction des modes de connection.	A-7

P L A N C H E S - J O I N T E S

(Numérotées par Chapitre)

- 4-1 Applications sémantiques des mots en programme
- 4-2 Mots fonctions auxiliaires (familles et applications sémantiques).

- 5-1 Ordinogramme global P1 (M01)
Démarrage, préparation lecteur, calage des origines
- 5-2 Ordinogramme global P3 (M01)
Rupture de séquence en fin d'usinage d'un bloc
- 5-3 Ordinogramme de la programmation manuelle
en usinage automatique SP1 (M01)
- 5-4 Ordinogramme de la lecture de un caractère en usinage automatique SP2 (M01)
- 5-5 Ordinogramme de la mise en mémoire d'un mot SP3 (M01)
- 5-6 Ordinogramme de mise en place de la bande SP6-SP7 (M01)
- 5-7 Ordinogrammes des ruptures de programme en mise en place bande SP8-SP9 (M01)
- 5-8 Ordinogrammes de la lecture de un caractère en avance SP10 (M01) en retour SP11 (M01)
- 5-9 Ordinogramme des arrêts extérieurs SP12 (M01)

- 7-1 Unité de lecture de un caractère - Plan de principe des liaisons.

BIBLIOGRAPHIE

- [1] ARSAC J. ; LENTIN A. ; NIVAT M. ; NOLIN L.
Algol Théorie et Pratique
Gauthier Villars PARIS 1965
- [2] BARTEE ; LEBOW ; REED
Theory and design of digital machines
Mac Graw Hill 1962
- [3] BERGE C.
Théorie des graphes et applications
Dunod 1967.
- [4] CHU Y.
Digital Computer design fundamentals
Mac Graw-Hill 1962
- [5] DEBRAINE P.
Machines et traitement de l'information
Masson 1967
- [6] FLORINE
La synthèse des machines logiques et son automatisation.
Dunod 1964.
- [7] FLORINE
Automatismes à séquences et commandes numériques
Dunod 1969.
- [8] HARARY
Graphe theory
Addison-Wesley 1969.
- [9] HARTMANIS et STEARNS
Algebraic Structure theory of sequential machines
Prentice Hall.

- [10] KAUFMANN
Méthodes et modèles de la recherche opérationnelle
Dunod 1968
- [11] PERRIN DENOUETTÉ DACLIN
Systèmes logiques
Dunod 1967
- [12] RAYMOND
Les principes des ordinateurs
Presses Universitaires 1969
- [13] SIMON
La commande numérique des machines outils
Eyrolles 1967
- [14] THILLIEZ
La commande numérique des machines
- [15] WILSON
Numerical control in manufacturing
Mac Graw-Hill 1963.
- [16] S D S . Silicon Logic Circuit modules
Documentation Scientific Data Systems
Réf.: 64-55-06A (Rev.3)
- [17] S D S Module Reference Manual
Documentation Scientific Data Systems
Ref. : 64-55-14 B
- [18] Normes Françaises
NF Z 62-010
NF Z 68-010, 030 ; 031 ; 032 ; 033.

INTRODUCTION

Dans le cadre du Laboratoire d'Automatique industrielle des ARTS et METIERS, nous avons étudié et réalisé une commande numérique de Machine-outil. Cette machine est relativement simple :

- . Elle génère les trajectoires à usiner par l'intermédiaire d'un calculateur d'interpolation de trajectoires linéaires et circulaires dans le plan X, Y.

- . Elle est programmée en usinage automatique par les données d'une bande programme écrite en langage codé machine-outil. En usinage semi-automatique les données sont affichées manuellement à l'aide de rotacteurs.

L'étude de cette machine nous a permis d'aborder et de résoudre des difficultés essentielles théoriques et technologiques que l'on rencontre lors de la conception d'une machine logique câblée du type industriel. Rappelons que ce sont des machines comportant un nombre élevé de fonctions logiques, d'entrées et de cycles de fonctionnement.

Afin de résoudre ces difficultés, nous avons été conduit, au fur et à mesure de la conception, à appliquer successivement différentes méthodes de raisonnement qui permettent d'établir le plus logiquement possible une machine.

Partant d'un cahier des charges relativement peu précis, indiquant les caractéristiques essentielles, formuler l'énoncé du problème, est la première difficulté. Il n'est pas pensable de rédiger cet énoncé en langage "de tous les jours", ni d'établir directement toutes les petites fonctions de détail de la machine. Nous avons effectué des analyses et des synthèses successives de l'ensemble des données ; finalement, nous sommes arrivé aux ordinogrammes détaillés définissant toutes les évolutions possibles de la machine, au cours de tous les cycles, en fonction des données du programme et des actions manuelles de l'opérateur, c'est à dire des entrées.

. Dans un premier temps, nous avons déterminé les propriétés syntaxiques d'un langage général machine-outil à l'aide de la méthode de Naur et de Backus.

Cette étude conduit à la synthèse du langage. Plutôt que d'avoir à travailler sur des propositions détaillées par mots, blocs et caractères, nous travaillerons sur des familles de propriétés syntaxiques identiques.

. Dans un deuxième temps, nous avons établi les ordinogrammes de cette machine, c'est à dire l'ensemble de ses applications sémantiques et les enchainements entre elles, en fonction des données du langage machine-outil et des actions manuelles de l'opérateur. Nous n'avons pas effectué une synthèse directe ni une analyse détaillée suivie d'une synthèse. Partant des propositions essentielles connues de manière très floue, nous avons établi les ordinogrammes globaux. Puis nous avons repris chaque proposition, nous l'avons détaillée; ainsi de proche en proche, nous sommes arrivé aux évolutions élémentaires correspondant à des applications sémantiques de la machine. Nous nous sommes arrêté à partir de l'instant, où pour détailler plus, il était nécessaire de connaître les fonctions logiques binaires qui réaliseront ces évolutions élémentaires.

L'énoncé logique est alors établi. C'est le travail de base qui, à notre avis, est le plus important. A toute omission correspondra la panne plus grave, la plus délicate à dépanner (en logique câblée). Elle peut imposer lors de la mise au point final, la modification complète d'une partie de la machine.

La deuxième famille de problèmes est essentiellement technologique, nous ne l'aborderons pas au cours de cette thèse, bien que ce soient des problèmes primordiaux quant au fonctionnement final de la machine. Précisons rapidement :

- Qu'il faut choisir un standard d'opérateurs logiques compatible avec la machine, en tenant compte de la rapidité des circuits et des conditions d'environnement (bruits électriques, vibrations mécaniques, et température ambiante).

- Qu'il est nécessaire de prévoir des zones dans la machine, attribuées en fonction du niveau énergétique des opérateurs, des niveaux de bruits tolérés et des problèmes thermiques. Cela conduit à séparer obligatoirement l'électrotechnique de puissance, la logique à relais, la logique électronique, puis à prévoir des opérateurs d'interfaces permettant d'adapter les impédances, de découpler les masses et de limiter la propagation des bruits.

Nous avons alors à notre disposition l'énoncé du problème, le standard des opérateurs de logique utilisables, il reste à établir les équations de la machine, les plans de principes, et les plans de câblage.

Il ne faut pas se lancer un peu au hasard, dans l'étude de petits sous-ensembles, à priori évidents, en espérant que l'on réussira bien à faire fonctionner la machine finale en inter-connectant le tout.

Il ne faut pas non plus avoir la prétention de déterminer en bloc toute la machine et d'effectuer des synthèses trop savantes qui conduiront à une machine logique très délicate à dépanner ou à modifier.

Nous déterminerons des unités fonctionnelles, c'est à dire des machines logiques réalisant une partie des évolutions prévues dans les ordinogrammes. Pour ce faire, nous rechercherons les boucles et les noeuds essentiels des ordinogrammes, et établirons une partition sur l'ensemble des évolutions prévues ; à une partie correspondra une unité fonctionnelle.

Les méthodes utilisées pour étudier les unités fonctionnelles sont classiques, c'est le type même des problèmes industriels. Pour établir une unité quelconque, il faut connaître les autres. Les entrées et les sorties sont mal définies, nombreuses, de plus, il est nécessaire d'en créer. La liberté est relativement grande, mais il faut cependant que l'unité obtenue respecte les évolutions et les enchainements prévus dans les ordinogrammes.

Partant de ces ordinogrammes, nous avons établi les graphes des séquences des états des sorties, en fonction des états des entrées. Insistons sur le fait que l'on ne connaît que l'état d'une partie des entrées, et uniquement aux époques où ces états impliquent des changements d'état des sorties. Il est donc nécessaire

d'établir un tableau des états des entrées, en commençant par l'état initial à la mise sous tension de l'unité et de suivre toutes les évolutions dans l'ordinogramme. Ceci établi, on est conduit au diagramme de fluence de l'unité, connu uniquement de par ses chemins non bouclés, c'est à dire les chemins d'évolution d'état interne en état interne.

Les équations logiques d'une unité (obtenues en tenant compte du standard des opérateurs de logique utilisables) la phase suivante est l'établissement des plans de principes détaillés. Rappelons qu'il est nécessaire d'élaborer le plus de variables auxiliaires communes à toutes les équations, et surtout de choisir les opérateurs en fonction de leurs caractéristiques électroniques (Réjection de bruit, temps de montée, temps de descente, temps de retard à la montée et à la descente, facteur pyramidal d'entrée : "Fan-in", facteur pyramidal de sortie : "Fan-out").

Ces plans obtenus, un travail de patience est nécessaire, il va de la mise en carte, au câblage de l'unité, en passant par tous les dossiers de repérage de chaque fil, de chaque carte, panier, armoire et câbles de liaisons.

Nous n'aborderons pas tous ces problèmes. Nous n'étudierons pas toute la machine, mais uniquement son unité d'entrée, en insistant sur les méthodes de conception logiques, sans aborder les problèmes technologiques.

Cette thèse comprend trois parties :

Dans la première partie, nous développerons différents problèmes théoriques de logique binaire que nous utiliserons dans la troisième partie, lors de l'étude des unités fonctionnelles. Nous établirons (chapitre 2) des propriétés remarquables des différentes machines logiques. Dans l'ordre :

- Nous préciserons les notions de pannes des entrées, d'initialisation à la mise sous tension.

- Nous énoncerons les propriétés des diagrammes de fluence (graphe aux évolutions d'état interne en état interne) de différents types de machines. Nous les appliquerons à l'étude des machines simples (fonctions mémoires et fonctions bascules) et nous en déduirons les équations aux chemins d'évolution.

Dans la deuxième partie, plutôt que d'étudier le langage particulier de la machine réalisée, nous établirons (chapitre 4) les propriétés syntaxiques du langage complet normalisé machine-outil. Cette étude nous permettra de déterminer (chapitre 5) les ordinogrammes d'une machine comportant toutes les propositions de ce langage.

Dans la troisième partie, après avoir éliminé (chapitre 5) les phases d'évolutions n'appartenant pas à la machine réalisée, nous rechercherons (chapitre 6) les unités fonctionnelles de l'unité d'entrée, puis nous établirons (chapitre 7 et 8) ces unités fonctionnelles.

2. MACHINES LOGIQUES DETERMINEES.

pages :

2-1- Généralités relatives aux machines logiques	9
2-1-1- Rappels relatifs aux signaux booléens.	9
2-1-2- Rappels relatifs aux machines de MOORE	10
2-1-3- Rappels relatifs aux machines de MEALY	11
2-2- <u>Diagramme de fluence des états internes. Notions de pannes et d'Initialisation.</u>	12
2-2-1- Introduction aux diagrammes de fluence des états internes	12
2-2-2- Discussions sur l'évolution d'un état interne q_i à un état interne q_j .	15
2-2-3- Notations simplifiées	19
2-2-4- Notion de panne des entrées	20
2-2-5- Notion d'initialisation à la mise sous tension	23
2-2-6- Conclusions.	25
2-3- <u>Etude des machines totalement déterminées.</u>	26
2-3-1- Définitions	26
2-3-2- Propriétés générales	26
2-3-3- Propriétés des chemins relatifs à un même sommet	27
2-3-4- Propriétés des trajets d'une partie de graphe	29
2-3-5- Propriétés des circuits élémentaires	31
2-4- <u>Application à l'étude des machines simples.</u>	34
2-4-1- Etude des mémoires dites de la 1ère espèce	34
2-4-2- Etude des fonctions bascules.	39

Introduction

Dans ce chapitre, après avoir rappelé rapidement les deux types classiques de machines logiques, nous étudierons les propriétés essentielles que doivent vérifier ces machines en fonctionnement normal, c'est-à-dire en fonction des états des entrées appartenant à E_G (ensemble des états générables des entrées).

Nous définirons les notions de diagramme de fluence ou plus exactement de graphe des évolutions des états internes, puis nous préciserons les notions d'évolutions sur panne des entrées et celles du problème primordial de l'initialisation d'une machine logique à la mise sous tension.

Ensuite nous étudierons plus en détail les propriétés que doivent vérifier les chemins du diagramme de fluence d'une machine pour que ce soit une machine totalement déterminée.

Nous étudierons ensuite deux machines de MOORE simples. La première sera une machine à un signal interne q_1 . La deuxième sera une machine à deux signaux internes q_1 et q_2 [un état interne étant interdit]. Nous établirons pour chaque opérateur la partition des états des entrées générables, telle que l'on puisse commander l'ensemble des évolutions possibles de ces opérateurs. Nous déduirons de ces partitions, les équations générales de chaque machine en fonction des parties non stables de chaque partition, puis les équations aux chemins d'évolutions [chemins stables exclus]. Finalement nous appliquerons ces résultats à l'étude d'une fonction mémoire simple et d'une bascule à niveau, utilisées dans la machine.

2-1 Généralités relatives aux machines logiques

Considérons une machine logique dont : l'ensemble des signaux booléens d'entrée est $\{E\}$, l'ensemble des signaux booléens de sortie est $\{S\}$, l'ensemble des signaux des variables internes est $\{Q\}$.

2-1-1 Rappels relatifs aux signaux booléens.

Nous allons rappeler par exemple pour l'ensemble des signaux d'entrée: $\{E\}$, les dénominations et notations essentielles

En remplaçant $\{E\}$ par $\{S\}$ ou $\{Q\}$, l'on retrouvera les dénominations et notations utilisées pour l'ensemble des signaux de sortie $\{S\}$ ou l'ensemble des signaux des variables internes $\{Q\}$.

Ensemble des signaux d'entrée, noté : $\{E\}$

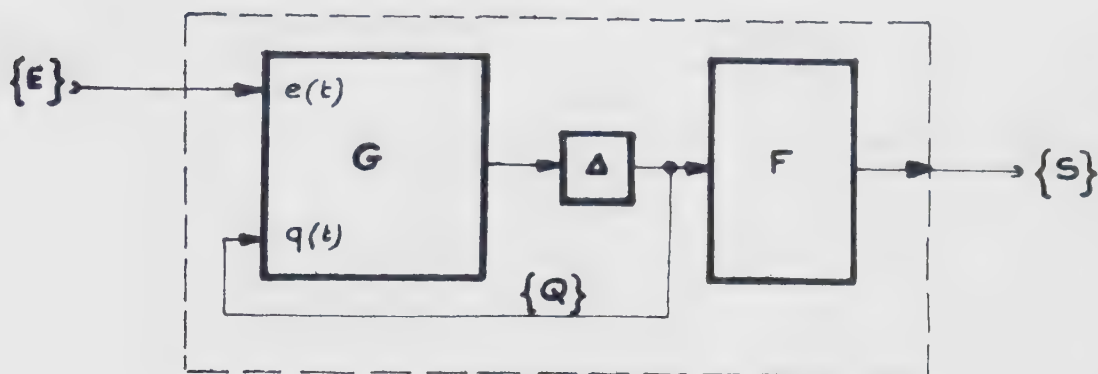
- E_S : Ensemble saturé des états des variables booléennes associées à l'ensemble des signaux booléens E .
- E_G : Ensemble des états générables de E .
- E_I : Ensemble des états interdits de E .
- E_j : le $j^{\text{ème}}$ sous-ensemble des états de E_S ;
 $E_j \subset E_S$
- $N E_i, \epsilon E_i, \delta E_i$: Respectivement, les partitions niveau, enclenchement, déclenchement des états de E_i
- $N a_i, \epsilon a_j^i, \delta a_k^j$: Respectivement des états particuliers niveau, enclenchement, déclenchement de E_S .

Ensemble des signaux de sortie, noté : $\{S\}$

Ensemble des signaux de variable interne, noté : $\{Q\}$

2-1-2 Rappels relatifs aux machines dites de Moore

Une machine dite de Moore se représente par le schéma suivant :



Dans une telle machine, l'on peut écrire qu'à une époque donnée t , le système étant à l'état interne $q_0(t) \in Q_G$, si l'on applique $e_1(t) \in E_G$

- a) $\{Q\}$ prendra à l'instant $(t + \Delta t)$, l'état $q_1 \in Q_G$ tel que :

$$(2-1) \quad \boxed{q_1(t + \Delta t) = G[q_0(t), e_1(t)]}$$

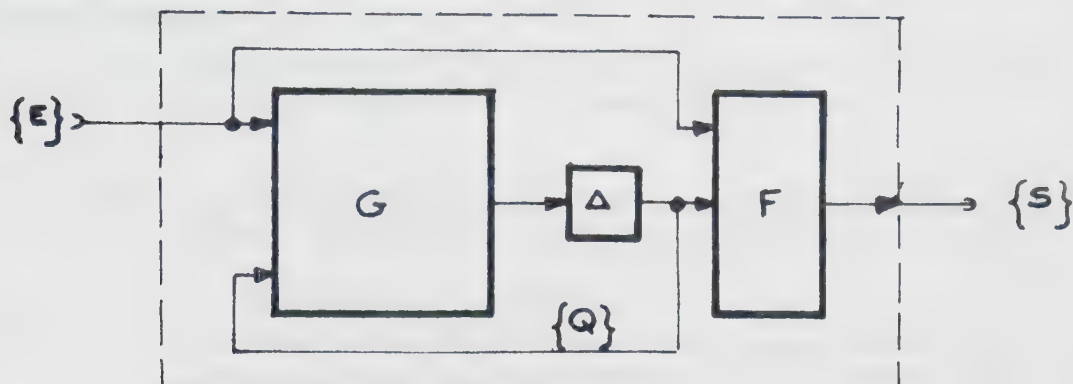
- b) $\{S\}$ étant à l'instant (t) , à l'état $s_0 \in S_G$, restera à cet état jusqu'à l'époque $(t + \Delta t)$, où il prendra l'état $s_1 \in S_G$, s_0 et s_1 étant tels que :

$$(2-2) \quad \boxed{\begin{aligned} s_0(t) &= F[q_0(t)] \\ s_1(t + \Delta t) &= F[q_1(t + \Delta t)] \end{aligned}}$$

Nous remarquons que dans une machine de Moore, l'ensemble des états des sorties n'est fonction que de l'ensemble des états des variables internes.

2-1-3 Rappels relatifs aux machines dites de Mealy

Une machine dite de Mealy se représente par le schéma suivant :



Dans une telle machine, l'on peut écrire qu'à une époque donnée (t) , le système étant à l'état interne $q_0(t) \in Q_G$, si l'on applique $e_1(t) \in E_G$:

- a) $\{Q\}$ prendra à l'instant $(t + \Delta t)$, l'état $q_1 \in Q_G$ tel que :

$$(2-3) \quad \underline{q_1(t + \Delta t) = G[q_0(t), e_1(t)]}$$

- b) $\{S\}$ étant à l'instant (t) à l'état $s_0 \in S_G$, prendra lorsque l'on applique $e_1(t) \in E_G$ l'état $s_1 \in S_G$ tel que :

$$(2-4) \quad \underline{s_1(t) = F[q_0(t), e_1(t)]}$$

et gardera cet état tant que dureront q_0 et e_1

- c) $\{S\}$ prendra à l'instant $(t + \Delta t)$, époque de l'apparition de $q_1(t + \Delta t)$, l'état $s_2 \in S_G$ tel que :

$$(2-5) \quad \underline{s_2(t + \Delta t) = F[q_1(t + \Delta t), e_1(t + \Delta t)]}$$

si par exemple e_1 a gardé son état de (t) à $(t + \Delta t)$.

Nous remarquons que dans une machine de Mealy, l'ensemble des états des sorties est fonction à chaque instant (t) de l'état, à cet instant, des signaux d'entrée et des signaux des variables internes.

2-2 Diagramme de fluence des états internes, notions de pannes et d'initialisation

Nous allons introduire dans ce paragraphe les graphes aux évolutions de ces machines, en nous limitant au graphe des évolutions d'état interne en état interne (dit diagramme de fluence). Nous n'étudierons que les machines logiques dont l'ensemble des variables internes n'agissent sur la machine considérée que de par leurs niveaux.

Puis nous aborderons le problème des pannes des entrées et celui de l' "initialisation" à la mise sous tension.

2-2-1 Généralités relatives aux diagrammes de fluence des états internes des machines de Moore et de Mealy

1°) Détermination du graphe

● Nous avons rappelé que pour chaque machine les états internes sont déterminés par la relation classique :

$$q'(t + \Delta t) = G[q(t), e(t)], \text{ d'après (3-1) et (3-3)}$$

● G est une application de $Q_G \times E_G \xrightarrow{G} Q_G$.
Convenons de représenter le graphe de cette application de la manière suivante, par :

a) les sommets : Chaque état $q_i \in Q_G$ est représenté, dans le plan (dit plan du graphe), par une circonférence, indexée par q_i , appelée sommet du graphe.

Exemple : 

b) les arcs : Il existe un arc reliant le sommet q_i au sommet q_j , dans cet ordre, si et seulement si, il existe : $e_k \in E_G$ tel que :

$$q_j(t + \Delta t) = G[q_i(t), e_k(t)]$$

Un tel état des signaux d'entrée, $e_k \in E_G$ est appelé ARC ; on le représente sur le graphe par un arc orienté de q_i à q_j , indexé par e_k .

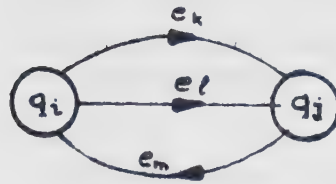
Exemple : 

c) les chemins : Etant donnés deux états q_i et q_j , l'on appelle chemin E_{ij} , l'ensemble des arcs reliant q_i à q_j dans cet ordre.

Nota : A chaque chemin E_{ij} correspond une partie de l'ensemble des états générables E_G des signaux d'entrée.

$$E_{ij} \subset E_G$$

d) Exemple de représentation.



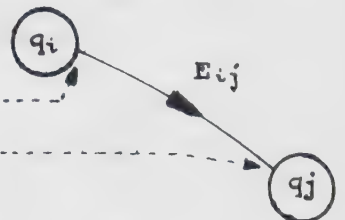
$\left. \begin{matrix} e_k \\ e_l \\ e_m \end{matrix} \right\}$ sont des arcs

$\left. \begin{matrix} E_{ij} = e_k \cup e_l \\ E_{ji} = e_m \end{matrix} \right\}$ sont des chemins

e) Terminologie particulière.

• Origine d'un chemin : -----

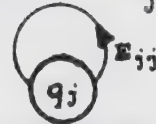
• Extrémité d'un chemin : -----



2°) Chemins particuliers

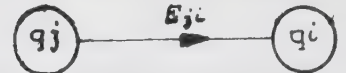
a) Les chemins bouclés : Nous appelons chemin bouclé, tout chemin reliant un sommet q_j à lui-même.

Nous le notons : E_{jj}



b) Les chemins incidents à un sommet : Nous appelons chemin incident à un sommet q_i , tout chemin reliant un sommet q_j différent de q_i , au sommet considéré q_i , dans cet ordre.

Exemple : E_{ji} est un chemin incident au sommet q_i



c) Les chemins émergents d'un sommet : Nous appelons chemin émergent d'un sommet q_i , tout chemin reliant ce sommet q_j à un sommet q_i différent, dans cet ordre.

Exemple : E_{ji} est un chemin émergent du sommet q_j .

3°) Les circuits d'un graphe

Un circuit est une partie de graphe telle que quel que soit le chemin considéré, son extrémité coïncide avec l'origine du chemin suivant par l'intermédiaire d'un sommet et son origine coïncide avec l'extrémité du chemin précédent par l'intermédiaire d'un sommet.

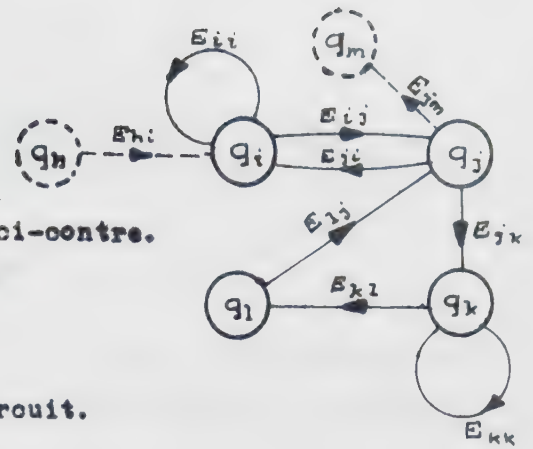
Exemple :

Soit la partie du graphe ci-contre.

Le circuit est représenté
en traits continus,

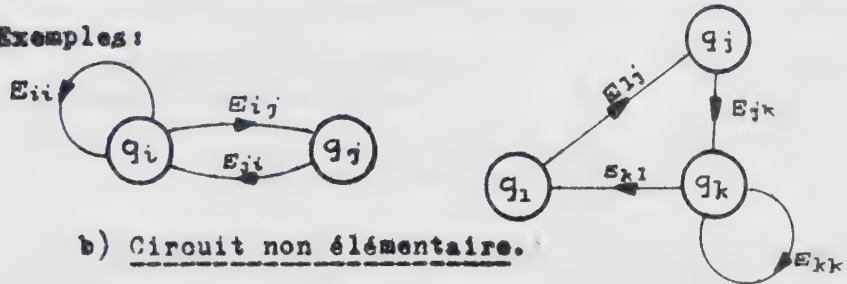
E_{hi} , E_{jm} , q_h , q_n

n'appartiennent pas au circuit.



a) Circuit élémentaire : Un circuit est dit élémentaire si tous ses sommets ne sont traversés qu'une fois lorsque l'on parcourt tous ses chemins (chemins bouclés exclus), en allant de l'origine d'un chemin quelconque à l'origine de ce même chemin.

Exemples :



b) Circuit non élémentaire.

4°) Les trajets

Un trajet est une partie de graphe telle

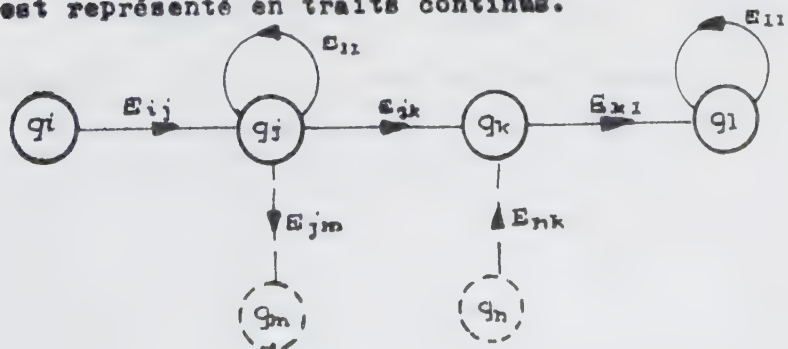
a) qu'il existe :

- Un sommet ne possédant pas de chemin incident dit sommet initial
- Un sommet ne possédant pas de chemin émergent dit sommet terminal.

b) que l'on puisse parcourir en allant du sommet initial au sommet terminal, tous les chemins de la partie du graphe, chemins bouclés exclus et en ne traversant qu'une fois chaque sommet de la partie de graphe, sommets initial et terminal exclus.

Exemple :

Soit la partie de graphe ci-dessous, le trajet est représenté en traits continus.



2-2-2 Discussions sur l'évolution d'un état interne q_i à un état interne q_j différent, q_i et $q_j \in Q_G$.

1°) Evolution naturelle de q_i à q_j .

Définition :

* L'évolution de q_i à q_j est dite naturelle si quel que soit l'état des signaux d'entrée le système évolue de q_i à q_j .

* L'application s'écrit :

$$\forall e_m \in E_G \quad (q_i, e_m) \longrightarrow (q_j)$$

que nous écrivons de manière plus condensée

$$(q_i) \longrightarrow (q_j)$$

* Le graphe de l'évolution de q_i à q_j est représenté alors comme suit :



* Le chemin E_{ij} est dit chemin naturel. Afin d'éviter toute erreur avec les autres chemins nous l'indexerons sur le graphe par $E_{ij} = 1$.

Théorème :

L'évolution d'un état de transition

(enclenchement ou déclenchement) ne peut se faire que vers l'état niveau suivant adjacent, quel que soit l'état des entrées.

$$(q_j^1) \longrightarrow q_j$$

Démonstration : Si $e_k \in E_G$, tel que :

$$(q_j^1, e_k) \longrightarrow q_{m \neq j}, \text{ cela impliquerait}$$

$$S \ q_j^1 = q_{m \neq j} \quad \text{or cela ne se peut.}$$

Le graphe se représente comme ci-dessous :



le chemin de q_j^1 à q_j est un chemin naturel.

2°) Evolution forcée vers un état q_j .

Définition :

L'évolution vers un état $q_j \in Q_G$ est dite forcée, si quel que soit l'état des signaux des variables internes, le système évolue vers q_j et y reste, pour tous les états des entrées appartenant à une partie E_j de E_G .

Le sommet q_j est dit sommet forcé.

Les chemins E_j sont dits chemins forcés.

L'application s'écrit :

$$\begin{cases} \forall q_i \in Q_G \cup Q_1 \\ \forall e_m \in E_j \subset E_G \end{cases} \longrightarrow (q_i, e_m) \longrightarrow (q_j)$$

Nous l'écrivons de manière condensée

$$(E_j) \longrightarrow (q_j)$$

Le graphe de l'évolution forcée vers q_j est représenté comme suit :



Le chemin E_j est un chemin incident à q_j .

Théorème 2

Si E_j est le chemin forcé d'un sommet q_j , l'ensemble des états de E_j est inclus dans le chemin bouclé E_{jj} de ce sommet

$$E_j \subset E_{jj}$$

Théorème 3

Si E_j est le chemin forcé d'un sommet q_j , l'ensemble des états de E_j est inclus pour chaque sommet $q_{i \neq j}$, dans l'ensemble des chemins émergents de ce sommet.

$$\forall q_{i \neq j} ; E_j \subset \bigcup_{k \neq i} E_{ik}$$

3°) Evolution entre deux états internes niveaux q_i et q_j consécutifs.

Rappelons que deux états niveaux q_i et q_j sont dits consécutifs si pour aller d'un état à l'autre un seul signal change de niveau.

• La succession des états internes entre deux niveaux consécutifs est q_i, q_j^i, q_j dans cet ordre puisque :

$$E q_i = P q_j = q_j^i$$

• L'application s'écrit, si E_{ij} est l'ensemble des états signaux d'entrée qui font évoluer de q_i à q_j^i :

$$\forall e_m \in E_{ij} \subset E_G \quad \begin{cases} (q_i, e_m) \longrightarrow (q_j^i) \\ (q_j^i, e_m) \longrightarrow (q_j) \end{cases}$$

D'après le théorème 1 il vient :

$$\forall e_m \in E_{ij} \subset E_G \quad \begin{cases} (q_i, e_m) \longrightarrow (q_j^i) \\ (q_j^i) \longrightarrow (q_j) \end{cases}$$

• Le graphe est alors représenté comme suit :



Nous le réduirons en éliminant conventionnellement l'état de transition q_j^i et le chemin naturel, il vient alors :



Théorème 4 Dans tout graphe à évolutions consécutives de niveau à niveau l'on peut éliminer l'ensemble des sommets correspondant à des états de transitions (enclenchement ou déclenchement)

4°) Evolution entre deux états internes niveaux q_i et q_j non consécutifs.

• Rappelons que deux états internes niveaux q_i et q_j sont non consécutifs si pour aller d'un état à l'autre il est nécessaire de changer d'état plusieurs signaux.

• La séquence d'évolution des états internes entre les deux niveaux q_i et q_j peut s'effectuer de plusieurs manières, en particulier :

a) En changeant d'état, successivement, un par un les signaux de manière à aller de q_i en q_j en n'ayant que des évolutions consécutives.

b) En changeant d'état "simultanément" tous les signaux qui doivent changer d'état.

• Soit B l'ensemble des signaux ayant le même état dans q_i et q_j , et A l'ensemble des signaux ayant des états différents. Changer d'état "simultanément" tous les signaux appartenant à A signifie commander, au "même instant", à tous ces signaux, de changer d'état.

• Chacun sait que les retards à la montée et à la descente sont aléatoires pour un ensemble d'opérateurs. Lorsque l'on commande à l'ensemble des signaux de A de changer d'état, l'indétermination des temps de retard implique que l'on génère de façon aléatoire l'ensemble des états possibles de A , autres que l'état des signaux de A dans q_i et dans q_j .

• L'ensemble des états généré lors de la transition de q_i à q_j sera dit ensemble des états aléatoires

Exemple :
$$\begin{cases} q_i = \bar{s}_1 \cdot s_2 \cdot \bar{s}_3 \cdot \bar{s}_4 \cdot s_5 \\ q_j = \bar{s}_1 \cdot s_2 \cdot s_3 \cdot s_4 \cdot s_5 \end{cases}$$

$$B = \{\bar{s}_1, s_2, \bar{s}_3\} ; \quad A = \{s_4, s_5\}$$

L'ensemble des états de A est représenté ci-contre :

$$\left\{ \begin{array}{cc} \bar{s}_4 & \bar{s}_5 \\ \bar{s}_4 & s_5 \\ s_4 & s_5 \\ s_4 & \bar{s}_5 \end{array} \right\}$$

L'ensemble des états aléatoires de A est : $\left\{ \begin{array}{cc} \bar{s}_4 & \bar{s}_5 \\ s_4 & s_5 \end{array} \right\} = A_{ij}$

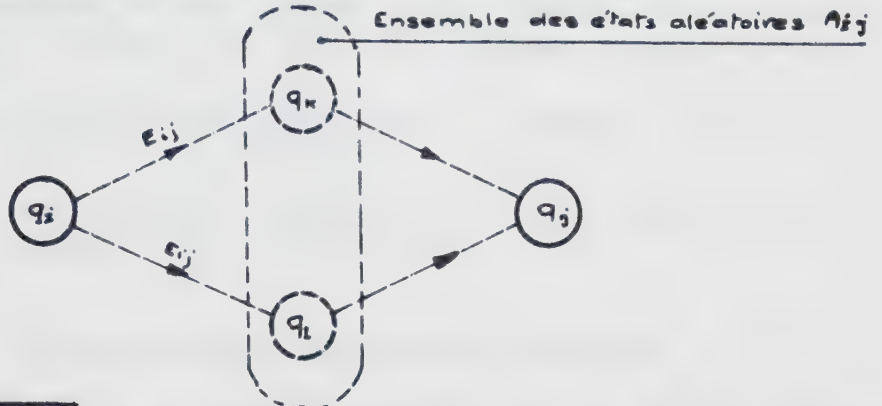
L'ensemble des états aléatoires généré lors de la transition de q_i à q_j sera donc :

$$q_k = \bar{s}_1 \cdot s_2 \cdot \bar{s}_3 \cdot \bar{s}_4 \cdot \bar{s}_5 \quad \text{et} \quad q_l = s_1 \cdot s_2 \cdot s_3 \cdot s_4 \cdot s_5$$

Le diagramme de fluence de transition de q_i à q_j se représentant comme suit :



est en fait l'un des trajets représenté sur le graphe des évolutions possibles aléatoires de q_i à q_j ci-dessous :



Théorème 5 Dans toute évolution entre deux états internes q_i à q_j non consécutifs, si E_{ij} implique le changement d'état "simultané" de toutes les variables qui doivent changer d'état, les variables internes décrivent un sous-ensemble d'états aléatoires, inclu dans l'ensemble des états aléatoires A_{ij} générables lors de la transition.

2-2-3 Notation simplifiée

Aucune erreur n'étant possible dans la formulation des équations de fonctionnement des machines de Moore ou de Mealy rappelée précédemment, nous conviendrons de les écrire le plus simplement possible de la manière suivante:

- Pour les machines de MOORE (équations de fonctionnement (3-1) et (3-2)), nous écrivons par exemple:

$$\begin{cases} q_0, e_1 \longrightarrow q_1 \\ q_1 \longrightarrow s_1 \end{cases}$$

- Pour les machines de MEALY, (équations de fonctionnement (3,3), (3-4), (3-5)), nous écrivons par exemple:

$$\begin{cases} q_0, e_1 \longrightarrow q_1 \\ q_0, e_1 \longrightarrow s_1 \\ q_1, e_1 \longrightarrow s_2 \end{cases}$$

Nous avons conservé les indices indiqués dans l'exemple des fonctions précédentes.

2-2-4 Notion de panne des entrées

Nous ne discuterons pas des pannes qui peuvent se produire dans la machine que nous étudions.

Nous ne considérerons que les possibilités d'évolution de la machine, d'état interne en état interne lorsque les entrées atteignent (par panne des machines élaborant ces entrées) :

- Soit des états appartenant à l'ensemble des états interdits des entrées $E_I \subset E_S$,
- Soit des états anormaux des entrées, lorsqu'elles se trouvent dans un état interne donné.

1°) Evolution de la machine pour les états $e_i \in E_I$

L'on peut concevoir plusieurs types de machines:

- Les machines à évolution pour les états interdits des entrées.
- Les machines à non évolution pour les états interdits des entrées.
- Les machines à blocage pour les états interdits des entrées.

a) Les machines à évolution pour les états interdits des entrées.

Ce sont des machines telles que :

$$\left[\begin{array}{ll} \exists q_1 \in Q_0 & \text{et} \quad \exists e_1 \in E_I \\ q_1, e_1 \longrightarrow q_j & q_1 \neq q_j \end{array} \right.$$

Proposition (1): Pour qu'une machine soit à évolution pour les états interdits des entrées, il suffit que tout ou partie de E_I soit inclu dans tout ou partie des chemins non bouclés du graphe des évolutions internes. (diagramme de fluence § 3-2-1)

b) Les machines à non évolution pour les états interdits des entrées.

Ce sont des machines telles que :

$$\left[\begin{array}{ll} \forall q_1 \in Q_0 & \text{et} \quad \forall e_1 \in E_I \\ q_1, e_1 \longrightarrow q_1 & \end{array} \right.$$

Proposition 2 : Pour qu'une machine soit à non évolution pour les états interdits des entrées il faut qu'aucun état interdit $\in E_I$ ne soit inclue dans aucun chemin non bouclé du graphe des évolutions internes.

c) Les machines à blocage pour les états interdits des entrées.

L'on peut concevoir une machine de détection de panne pour les états interdits $\in E_I$, notée DpE, d'états DpE et $\overline{\text{DpE}}$, telle que :

$$\left[\begin{array}{ll} e_1 \longrightarrow \text{DpE} & V_{e_1} \in E_I, \forall \text{l'état DpE ou } \overline{\text{DpE}} \\ \text{DpE}, e_j \longrightarrow \text{DpE} & V_{e_j} \in E_S \\ \overline{\text{DpE}}, e_k \longrightarrow \overline{\text{DpE}} & V_{e_k} \in E_G \\ \text{DpE}, (e_k \cdot \text{AcqDpE}) \longrightarrow \overline{\text{DpE}} & V_{e_k} \in E_G \end{array} \right.$$

AcqDpE sera dit acquittement de la détection de panne DpE.

Les évolutions d'état interne en état interne seront telles que :

$$\left[\begin{array}{ll} q_1, (e_1 \cdot \text{DpE}) \longrightarrow q_1 & V_{q_1} \in Q_S ; V_{e_1} \in E_S \\ q_1, (e_1 \cdot \overline{\text{DpE}}) \longrightarrow q_j & \text{fonctionnement normal de la machine} \end{array} \right.$$

Proposition 3 : La machine sera à blocage sur panne des entrées si toutes les équations booléennes des chemins sont multipliées par $\overline{\text{DpE}}$. Dans ce cas tout ou partie des états interdits pourra être inclue dans tout ou partie des chemins du graphe des évolutions internes.

Nota : Rappelons que l'équation booléenne d'un chemin E_{ij} est telle que :

$$\begin{array}{ll} V_{e_1} \in E_{ij} ; * E_{ij} = 1 & \text{soit en notation booléenne } E_{ij} \\ V_{e_1} \notin E_{ij} ; * E_{ij} = 0 & \text{soit en notation booléenne } \overline{E_{ij}} \end{array}$$

D'où l'équation booléenne classique.

$$E_{ij} = \sum_{e_1 \in E_{ij}} e_1$$

2°) Evolution de la machine pour des états anormaux des entrées apparaissant lorsqu'elle se trouve dans un état interne donné.

a) l'on peut effectuer la synthèse globale de la machine en incluant toutes les séquences d'entrée possibles (les séquences conduisant au fonctionnement normal et les autres). Cela conduit à introduire des états internes et des chemins supplémentaires.

b) l'on peut concevoir dans la machine une unité d'alarme centralisée, notée $A\bar{L}c$ d'états niveaux $A\bar{L}c$ et $\overline{A\bar{L}c}$ tels que :

$$\begin{cases} q_1, (e_j \cdot A\bar{L}c) \longrightarrow q_1 & \forall q_1 \in Q_S ; \forall e_j \in E_S \\ q_1, (e_j \cdot \overline{A\bar{L}c}) \longrightarrow q_j & \text{c'est-à-dire le} \end{cases}$$

fonctionnement normal, par évolutions d'état interne en état interne prévu.

L'unité d'alarme sera une machine logique d'équations suivantes :

Si $Q_1 \subset Q_G$ et que quel que soit $q_1 \in Q_1$, il existe un sous-ensemble d'état d'entrée $E q_1$ tel que pour ce sommet donné, $E q_1$ soit anormal nous aurons :

$$[q_1, E q_1] \longrightarrow A\bar{L}c$$

la machine est alors bloquée en q_1 dit état interne d'alarme.

Si ensuite il apparaît en q_1 un état normal n'appartenant pas à $E q_1$, alors l'on peut acquitter $A\bar{L}c$ par l'acquiescement de l'alarme $AcqA\bar{L}c$.

$$[A\bar{L}c, [AcqA\bar{L}c \cdot q_1 \cdot \overline{E q_1}]] \longrightarrow \overline{A\bar{L}c}$$

Nota : Rappelons que

$AcqA\bar{L}c \cdot q_1 \cdot \overline{E q_1}$ est un sous-ensemble d'état de l'ensemble des signaux

$$\{AcqA\bar{L}c\} \times \{Q\} \times \{E\}$$

$\overline{E q_1}$ signifiant que les entrées $\{E\}$ ne sont pas dans un des états appartenant à $E q_1$.

Proposition 4 : Une machine à alarme centralisée sera à blocage sur état interne d'alarme si toutes les équations booléennes des chemins émergents d'un sommet $q_1 \in Q_1$ sont multipliées par $A\delta$.

c) l'on peut concevoir des machines à unité d'alarme plus évoluée, par exemple des machines à un état d'alarme $A\delta q_1$ par état $q_1 \in Q_1$, à blocage éventuel sur l'état alarme q_1 considéré.

2-2-5 Notion d' "initialisation" à la mise sous tension d'une machine logique.

a) Nous admettrons que pendant la phase de mise sous tension d'une machine logique tous les signaux ont un état quelconque (les variables internes, les sorties et les entrées élaborées par d'autres machines qui sont en phase de mise sous tension ou qui ne le sont pas encore).

b) Etudier les évolutions de la machine ou encore mieux de l'ensemble des machines interconnectées et prévoir, dans quel état total elles se stabiliseront, si elles se stabilisent, est impossible.

c) La mise à un état total initial d'une machine logique se fera en trois phases :

- mise sous tension et envoi d'un ordre MZG (état 1)
- mise à l'état initial total par l'ordre MZG
- début des évolutions prévues de la machine aussitôt que MZG prend l'état 0 ($\overline{\text{MZG}}$)

L'ordre MZG ne sera pas classé dans l'ensemble des entrées $\{E\}$. Nous définirons un nouvel ensemble des signaux d'entrée : $\{E\} \cup \text{MZG}$. Afin de déterminer les états de ce nouvel ensemble nous utiliserons la notation suivante :

Par exemple si $\{E\}$ est à l'état e_3 nous noterons :

$$\left[\begin{array}{ll} (\text{MZG} . e_3) & \text{si MZG est à l'état un} \\ e_3 & \text{si MZG est à l'état zéro } (\overline{\text{MZG}}) \end{array} \right.$$

afin d'alléger les écritures lors de l'étude de l'évolution de la machine pendant le fonctionnement.

d) Méthode d' "initialisation" d'une machine à un seul état initial total possible par exemple : e_1, q_2, s_2 à l'époque t_0^+ .

t_0 étant l'époque de déclenchement de MZG ($\overline{\text{MZG}}$)

• A l'époque t_0^+ l'évolution commence, nous avons.

$$\begin{cases} q_2, e_1 \longrightarrow q_j & \begin{cases} q_2 \text{ et } q_j \in Q_G \\ e_1 \in E_G \end{cases} \\ \text{et } q_2 \longrightarrow s_2 & \text{ou } (q_2, e_1) \longrightarrow s_2 ; s_2 \in S_G \end{cases}$$

suyvant que c'est une machine de MOORE ou de MEALY.

• Méthode d' "initialisation forcée" avant l'instant t_0
c'est-à-dire pendant l'époque t_0^-

"Initialisation" des variables internes.

$$q_1, (MZG \cdot e_1) \longrightarrow q_2 \quad \forall q_1 \in Q_S \quad \text{et} \quad \forall e_1 \in E_S$$

c'est une évolution forcée par MZG.

Nota : l'on ne choisit pas la solution d'initialisation de toutes les entrées $\in \{E\}$ pour des raisons de coût.

"Initialisation" des sorties d'une machine de MOORE

$$q_2 \longrightarrow s_2$$

"Initialisation" des sorties d'une machine de MEALY

$$q_2, (MZG \cdot e_1) \longrightarrow s_2 \quad \forall e_1 \in E_S$$

c'est une initialisation forcée par MZG (dans le circuit combinatoire) au plus près des sorties.

"Initialisation" des entrées.

Aucun problème ne se pose pour la partie des signaux d'entrée $\{A\}$ qui sont des sorties de machines initialisées par MZG à l'état e_1^A sous réserve évidemment que l'état de chaque signal (sortie $\longrightarrow$ entrée) soit identique.

Reste le cas des entrées $\{B\}$ qui ne sont pas encore initialisées et qui ne le seront qu'après l'époque t_0^+ .

Si $\{B\}$ est la partie des signaux non initialisés à l'époque t_0 , lors de leur initialisation après t_0^+ ils généreront l'ensemble des états de $\{B\}$ c'est-à-dire $B = \Sigma e_j^B \quad \forall e_j^B \in B$

L'ensemble des entrées $\{E\}$ généreront l'ensemble des états suivants : $e_1^A \cdot (\Sigma e_j^B) \quad \forall e_j^B \in B$ dans lequel est inclu e_1 état initial de E .

Si il existe un état $e_1^A \dots e_j^B$ tel que

$$q_2, (e_1^A \dots e_j^B) \longrightarrow q_j \neq q_2$$

la machine évolue sur les aléas des entrées de la partie $\{B\}$ (pendant l'initialisation) or cela ne se peut d'où :

$$\forall e_j^B \quad q_1, (e_1^A \dots e_j^B) \longrightarrow q_2$$

et en particulier pour l'état $e_1 \in e_1^A \dots (e_j^B)$

$$\text{c'est-à-dire que} \quad q_2, e_1 \longrightarrow q_2$$

e) Conclusions

Il faut effectuer à la mise sous tension d'une machine logique sa mise à l'état initial total (par exemple : e_1, q_1, s_1) par un ordre MZG.

Tant que MZG est à l'état 1, la machine ne peut pas évoluer. Elle est stable sur son état interne initial forcé q_1

$$q_1, (MZG \cdot e_1) \longrightarrow q_1 \quad \forall q_1 \in Q_S, \forall e_1 \in E_S.$$

Dans les machines de MEALY il faut effectuer la mise à l'état initial des sorties s_1

$$q_1, (MZG \cdot e_1) \longrightarrow s_1 \quad \forall e_1 \in E_S$$

sauf si les entrées sont à l'état initial e_1 tel que

$$q_1, e_1 \longrightarrow s_1.$$

Dans le cas de machine logique interconnectée et à mise aux conditions initiales forcées différentes nous admettrons que toutes les machines initialisées en 1er ne pourront évoluer que lorsque les autres machines seront initialisées.

2-2-6 Conclusions

L'étude précédente montre que la machine logique se décompose en :

a) une machine dite de fonctionnement prévue par évolutions en fonction uniquement des états internes $q_1 \in Q_G$ et des états des entrées $e_j \in E_G$ (ensemble des états générables tel que $E_G = E_S - E_I$)

L'ensemble des états interdits sera affecté dans les chemins comme nous avons précisé dans les propositions 1, 2 ou 3 en fonction du choix effectué.

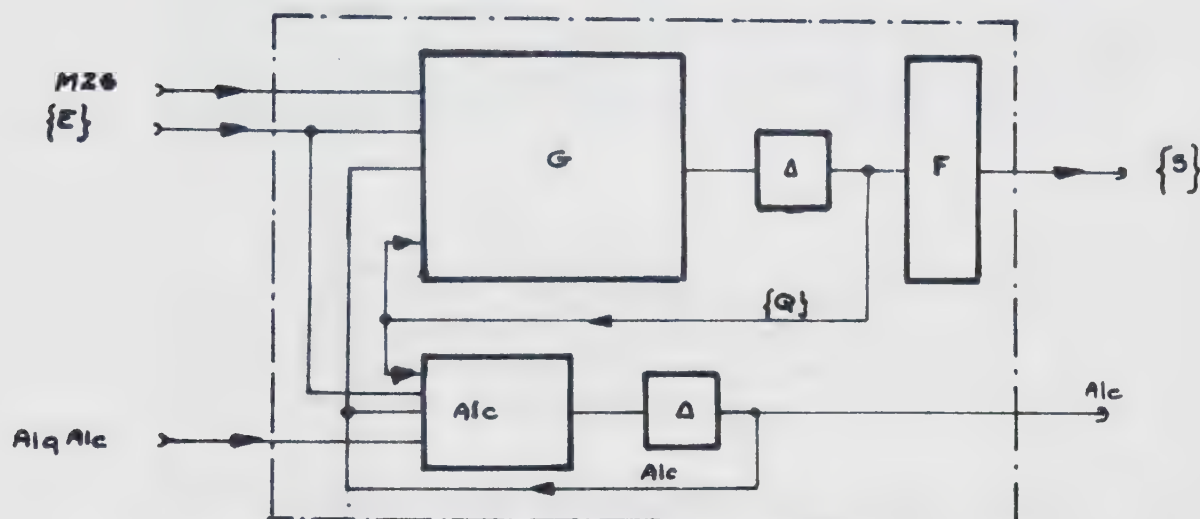
Nous remarquerons que les signaux $DpE, A/c, \{A/q_1\}, MZG$ bien qu'étant des entrées de la machine logique seront considérés comme n'appartenant pas à l'ensemble des entrées $\{E\}$ dit ensemble des entrées de fonctionnement.

b) Une machine (ou plusieurs) de détection de panne et, ou d'alarme, déterminée précédemment dont les états des sorties $[DpE, A/c, A/q_1]$ seront affectés dans les équations

booléennes des chemins du diagramme de fluence comme il a été indiqué précédemment.

Nous remarquerons que cette deuxième machine est une machine de MOORE ou de MEALY (dans le cas de machines d'alarme complexes)

Schématisons une machine à alarme centralisée et à non évolution pour les états interdits des entrées.



2-3 Etude des machines totalement déterminées

2-3-1 Définition

Nous appellerons machine logique totalement déterminée une machine logique telle que, à chaque instant $t \in T$, l'on connaît l'état de toutes les sorties et de toutes les variables internes, en fonction des séquences des états des entrées, appliquées jusqu'à cet instant.

2-3-2 Propriétés générales

1*) Une machine totalement déterminée est un système déterministe : aux mêmes causes (mêmes séquences des états d'entrée et mêmes conditions initiales des variables internes et des sorties) correspondent les mêmes effets (mêmes séquences des états des variables internes et des sorties).

2*) **Théorème 6** Une machine totalement déterminée est une machine à évolutions d'états internes consécutifs. Son diagramme de fluence est à évolutions consécutives, c'est-à-dire que quel que soit le chemin non bouclé E_{ij} ($i \neq j$) considéré appartenant au graphe, il relie deux sommets du graphe dont les états sont consécutifs.

En effet si il existe dans un diagramme de fluence deux sommets q_i et q_j non consécutifs reliés par un chemin E_{ij} , soit :

- l'on rend l'évolution de q_i à q_j consécutive et l'on ramène le graphe à un graphe consécutif.

- l'on convient de changer l'état, simultanément de plusieurs signaux. Dans ce cas l'on décrit de façon aléatoire des états appartenant à l'ensemble des états aléatoires A_{ij} générable lors de l'évolution de q_i à q_j . Il existe donc une époque pendant laquelle l'on ne connaît pas les états internes de la machine, celle-ci n'est donc pas totalement déterminée.

3°) Nous allons rechercher les propriétés que doivent vérifier les chemins, trajets et circuits d'un diagramme de fluence d'une machine logique pour que celle-ci soit une machine totalement déterminée, après avoir exclu de tout chemin, tout état appartenant à E_I .

2-3-3 Propriétés des chemins relatifs à un même sommet

Recherchons les propriétés que doivent vérifier l'ensemble des chemins incidents, émergents et bouclés, reliés à un même sommet, quel que soit ce sommet appartenant au graphe.

1°) Propriétés entre chemins émergents

Théorème 7 Quel que soit le sommet q_j considéré appartenant au graphe, aucun état des signaux d'entrée ne peut appartenir à deux chemins émergents différents de ce sommet.

Démonstration :

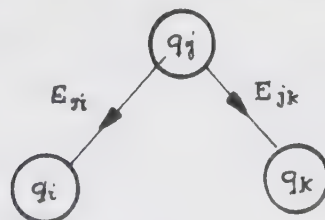
Si $\exists e_m \in E_{ji}$ et $e_m \in E_{jk}$ avec $i \neq j \neq k$

$$\left. \begin{array}{l} e_m \in E_{ji} \longleftrightarrow (q_j, e_m) \longrightarrow q_i \\ e_m \in E_{jk} \longleftrightarrow (q_j, e_m) \longrightarrow q_k \end{array} \right\} \text{ avec } q_i \neq q_k$$

Cela ne se peut, la machine ne serait pas déterministe.

Nota : Les chemins émergents d'un même sommet sont deux à deux disjoints.

$$\left[\begin{array}{l} \forall e_m \in \bigcup_{i \neq j} E_{ji} \\ E_{ji} \cap E_{jk} = \emptyset \\ \forall i \text{ et } k \end{array} \right.$$



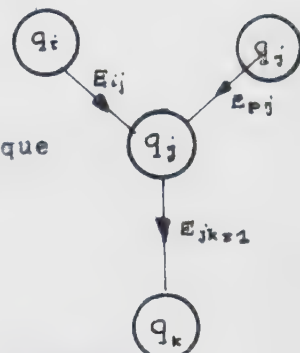
Corollaire 1 S'il existe un chemin émergent naturel $E_{jk} = 1$ d'un sommet q_j du graphe alors :

C'est le seul chemin émergent de ce sommet.

Démonstration :

$\forall q_j \in Q_G$ tel que $\exists E_{jk} = 1$, cela implique

$$\begin{cases} E_{jj} = \emptyset \\ \bigcup_{k \neq j} E_{jk} = \emptyset \end{cases}$$



Corollaire 2 Si dans un graphe, il existe un sommet forcé q_j de chemin forcé E_j , quel que soit $q_i \neq q_j$; E_j est inclus dans la réunion des chemins émergents de q_i .

$$E_j \subset \bigcup_{k \neq j} E_{ik}, \quad \forall q_i \in Q_G - q_j$$

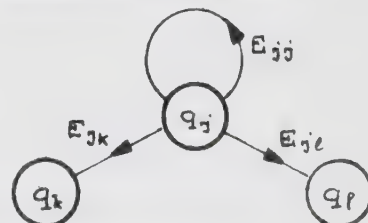
Corollaire 3 Si dans un graphe il existe plusieurs sommets forcés $q_j, \dots, q_k$ de chemins forcés $E_j, \dots, E_k$ respectivement, tous ces chemins forcés sont disjoints deux à deux.

$E_j \cap E_k = \emptyset, \quad \forall E_j, E_k$ appartenant à l'ensemble des chemins forcés du graphe.

2°) Propriétés entre chemins émergents et chemins bouclés

Théorème 8 Quel que soit le sommet q_j considéré du graphe, l'ensemble des états du chemin bouclé E_j de ce sommet est le complément de l'ensemble des états de tous les chemins émergents E_{jk} ($k \neq j$) de ce sommet, dans l'ensemble des états générables E_G des signaux d'entrée.

$$\begin{cases} \forall q_j \in Q_G \\ E_{jj} = E_G - \bigcup_{k \neq j} E_{jk} \end{cases}$$



Démonstration :

Si $\exists e_k \in E_{jj}$ et $e_k \notin E_G - \bigcup_{k \neq j} E_{jk}$
alors $e_k \in \bigcup_{k \neq j} E_{jk}$

$$e_k \in E_{jj} \iff (q_j, e_k) \longrightarrow q_j$$

$$e_k \in E_{jk} \iff (q_j, e_k) \longrightarrow q_k$$

q_j étant différent de q_k , s'il existait e_k vérifiant notre supposition, la machine ne serait pas déterministe.

Corollaire 4 S'il existe un chemin émergent naturel $E_{jk} = 1$ d'un sommet q_j du graphe, quel que soit ce sommet, alors, il n'existe pas de chemin bouclé E_{jj} .

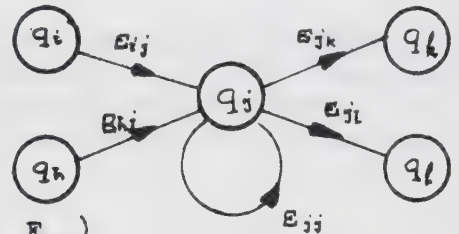
Corollaire 5 Si dans un graphe, il existe un sommet forcé q_j , de chemin forcé E_j , alors, quel que soit le chemin bouclé $E_{ii} \neq E_{jj}$,
 $E_{ii} \cap E_j = \emptyset \quad \forall i \neq j.$

3°) Propriétés entre chemins incidents et émergents

Corollaire 6 S'il existe un ensemble d'états des entrées $E_1 \subset E$, appartenant aux chemins incidents et émergents d'un sommet q_j du graphe, quel que soit $e_m \in E_1$, alors :

- e_m n'appartient qu'à un seul chemin émergent
- e_m n'appartient pas au chemin bouclé E_{jj} de q_j
- e_m peut appartenir à tout ou partie des chemins incidents à q_j .

Schéma et formulation



$$\text{Si } E_1 = \left(\bigcup_{i \neq j} E_{ij} \right) \cap \left(\bigcup_{k \neq j} E_{jk} \right)$$

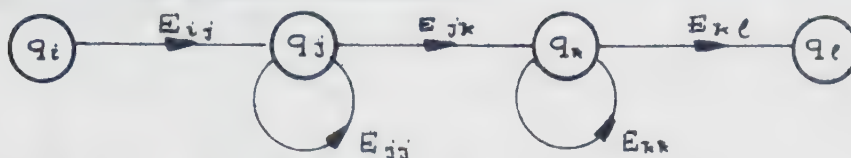
$$\text{Alors } \forall e_m \in E_1 \longrightarrow \begin{cases} e_m \in \bigcup_{i \neq j} E_{ij} & \forall i \neq j \\ e_m \notin E_{jj} \\ \exists k \neq j \text{ tel que } e_m \in E_{jk} \end{cases}$$

2-3-4 Propriétés des trajets d'une partie de graphe

La partie de graphe considérée est un trajet de sommet initial q_1 et de sommet final q_g .

Théorème 9 S'il existe un ensemble d'états niveau $N E_1$, appartenant à tous les chemins d'un trajet, chemins bouclés exclus, pour tout état $e_m \in N E_1$, l'état des variables internes évolue vers le sommet final du trajet tant que dure e_m .

Démonstration : Soit le trajet ci-dessous partie d'un graphe consécutif.



Si $\exists N E_1 = E_{ij} \cap E_{jk} \cap E_{kl} \cap \dots \cap$

Le théorème (8) $\implies N E_1 \not\subset E_{jj} \cup E_{kk} \dots$

Le long du trajet l'application s'écrit, $\forall e_m \in N E_1$,

$$\begin{aligned}
 (q_i, e_m) &\longrightarrow q_j^i \longrightarrow q_j \\
 (q_j, e_m) &\longrightarrow q_k^j \longrightarrow q_k \\
 (q_k, e_m) &\longrightarrow q_l^k \longrightarrow q_l
 \end{aligned}$$

Si Δt est le temps de retard moyen d'une évolution d'un état au suivant, le temps de parcours du trajet est $n \cdot \Delta t$ si n est le nombre de chemins, l'état e_m doit durer plus que $n \Delta t$ si l'on veut évoluer du point initial au point final du trajet sous e_m .

Théorème 10 S'il existe un ensemble d'états de transitions (enclenchements, ou déclenchements) : $\varepsilon E_1 \cup \delta E_1$ appartenant à tous les chemins d'un trajet, chemins bouclés exclus, pour tout état $e_m^m \in \varepsilon E_1 \cup \delta E_1$, l'état des variables internes n'évolue que d'un seul état.

Démonstration : Considérons le trajet ci-dessus, du théorème (9).

Si $\exists (\varepsilon E_1 \cup \delta E_1) \subset E_{ij} \cap E_{jk} \cap E_{kl} \cap \dots$

• Le théorème (8) $\implies \varepsilon E_1 \cup \delta E_1 \not\subset E_{jj} \cup E_{kk} \cup \dots$

• $\forall e_n^m \in (\varepsilon E_1 \cup \delta E_1)$ l'état des entrées adjacent suivant est $S e_n^m = e_n$, l'application $(e_n^m) \longrightarrow e_n$ est instantanée.

• L'application le long du trajet s'écrit

$$\begin{aligned}
 (q_i, e_n^m) &\xrightarrow{\Delta t} q_j^i \longrightarrow q_j \\
 (q_j, e_n^m) &\longrightarrow q_k^j \dots
 \end{aligned}$$

Si les entrées en q_j étaient à nouveau à l'état e_n^m , elles auraient au moins décrit la séquence suivante :

$$e_n^m \longrightarrow e_n \longrightarrow e_m^n \longrightarrow e_m \longrightarrow e_n^m$$

et l'état e_n^m serait apparu deux fois et non une seule.

Corollaire 7 Etant donné un graphe, les évolutions vers un sommet q_j donné seront stables en q_j pour des états des entrées si :

$$\left[\begin{array}{l} \forall e_m \in \bigcup_{i \neq j} N E_{ij}, \quad e_m \in E_{jj} (\longleftrightarrow e_m \notin \bigcup_{k \neq j} N E_{jk}) \\ \forall e_n^m \in \bigcup_{i \neq j} (\epsilon E_{ij} \cup \delta E_{ij}), \quad e_n = S e_n^m \in E_{jj} (\longleftrightarrow \\ \quad e_n = S e_n^m \notin \bigcup_{k \neq j} N E_{jk}) \end{array} \right.$$

Nota : Un état interne q_j ayant un chemin émergent naturel $E_{jk} = 1$ ne peut être stable pour aucun état des entrées.

Corollaire 8 Etant donné un graphe, les évolutions vers un sommet q_j donné sont instables en q_j pour tous les états des entrées si :

$$\left[\begin{array}{l} \forall e_m \in \bigcup_{i \neq j} N E_{ij}, \quad e_m \in \bigcup_{k \neq j} N E_{jk} \\ \forall e_n^m \in \bigcup_{i \neq j} (\epsilon E_{ij} \cup \delta E_{ij}), \quad e_n = S e_n^m \in \bigcup_{k \neq j} N E_{jk} \end{array} \right.$$

Dans ce cas, tout état $e_i \notin \bigcup_{i \neq j} E_{ij}$ et $\in \bigcup_{k \neq j} E_{jk}$ est un état à effet nul sur le système.

Corollaire 9 Etant donnés deux sommets q_i et q_j d'un graphe, reliés par le chemin E_{ij} , q_i est inaccessible depuis q_j si :

$$\left[\begin{array}{l} \bigcup_{h \neq i} N E_{hi} = \bigcup_{j \neq i} N E_{ij} - E_{ij} \\ \bigcup_{h \neq i} (\epsilon E_{hi} \cup \delta E_{hi}) = \bigcup_{j \neq i} N E_{ij} - E_{ij} \end{array} \right.$$

Nota : Un tel chemin E_{ij} ne peut exister dans une machine logique.

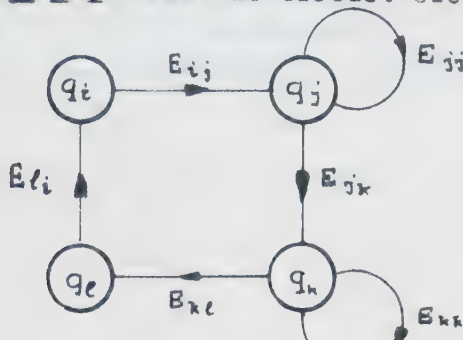
2-3-5 Propriétés des circuits élémentaires

Considérons une partie de graphe qui est un circuit élémentaire.

Si nous remarquons qu'un circuit élémentaire est un trajet auquel on a ajouté un chemin (appartenant au graphe) reliant le sommet final au sommet initial de ce trajet, il vient :

Théorème 11 Il ne peut pas exister un ensemble d'états niveau $N E_1$, appartenant à tous les chemins d'un circuit élémentaire, chemins bouclés exclus.

Démonstration : Soit le circuit élémentaire ci-dessous



$$\text{Si } N E_1 = E_{ij} \cap E_{jk} \cap E_{kl} \dots \cap E_{li} \neq \emptyset$$

- Le théorème (8) $\Rightarrow N E_1 \neq E_{jj} \cup E_{kk} \dots$

- L'extension du théorème (9) implique que $\forall e_m \in N E_1$, l'évolution de sommet en sommet dure tant que dure e_m . Lorsque l'on cesse d'appliquer l'état e_m le système se trouve en un sommet du circuit élémentaire quelconque, imprévisible, la machine n'est pas déterministe.

Théorème 12 S'il existe un ensemble d'états de transitions (enclenchements ou déclenchements) $\varepsilon E_1 \cup \delta E_1$, appartenant à tous les chemins d'un circuit élémentaire, chemins bouclés exclus, pour tout état $e_m \in \varepsilon E_1 \cup \delta E_1$, l'état des variables internes n'évolue que d'un seul état.

Démonstration : identique au théorème (10)

Considérons une partie de graphe qui est un circuit non élémentaire.

C'est un circuit tel qu'il existe au moins un sommet que l'on traverse au moins deux fois en parcourant tous les chemins, chemins bouclés exclus, de l'origine d'un chemin à cette même origine.

Théorème 13 Il n'existe aucun état des entrées appartenant à tous les chemins, chemins bouclés exclus, d'un circuit non élémentaire.

Démonstration :

Montrons qu'un circuit est un circuit non élémentaire s'il possède au moins un sommet q_i ayant deux chemins émergents distincts (E_{ij} et E_{ik}).

Si nous montrons cela le théorème 13 sera démontré car : si il existait un état des entrées appartenant à tous les chemins (chemins bouclés exclus) d'un circuit, il appar-

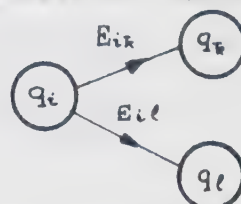
tiendrait à E_{ij} et E_{ih} . Or cela ne se peut car $E_{ij} \cap E_{ih} = \emptyset$ (théorème 7)

Considérons un circuit non élémentaire, il existe donc dans ce circuit au moins un sommet q_i que l'on traverse au moins deux fois en le parcourant.

Enumérons les configurations possibles des chemins liés à un tel sommet.

- a) Il existe au moins deux chemins émergents: d'après le théorème 7

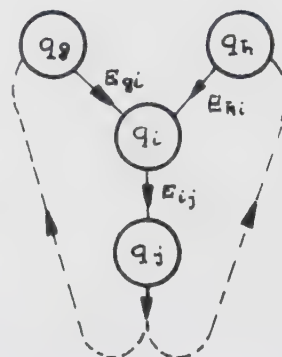
$$E_{ij} \cap E_{ik} = \emptyset \quad k \neq j \neq i$$



- b) Il existe au moins deux chemins incidents E_{gi}, E_{hi} et un seul chemin émergent E_{ij}

Le circuit est formé d'au moins deux circuits élémentaires distincts admettant E_{ij} comme chemin commun.

Il existe alors au moins un sommet q_m appartenant à ces deux circuits élémentaires ayant deux chemins émergents distincts appartenant respectivement aux deux circuits (sinon les deux circuits seraient confondus).



- c) Il existe un sommet q_i ayant un seul chemin incident E_{hi} et un seul chemin émergent E_{ij} parcouru au moins deux fois lorsque l'on parcourt le circuit non élémentaire. L'on montre de manière analogue au cas ci-dessus que le circuit possède au moins un sommet ayant deux chemins émergents.

2-4- Application à l'étude de machines simples :

2-4-1 Etude des mémoires dites de la première espèce

Considérons une machine de MOORE à une seule variable interne q , d'états niveaux $q_0 = \bar{q}$, et $q_1 = q$ et un ensemble d'entrées $\{E\}$, E_1 étant l'ensemble des états interdits de ces entrées.

Cette machine étant telle que l'état de sa sortie q à l'instant $t + \Delta t$ n'est fonction que de l'état de q et de l'état des entrées à l'époque t , indépendamment des séquences.

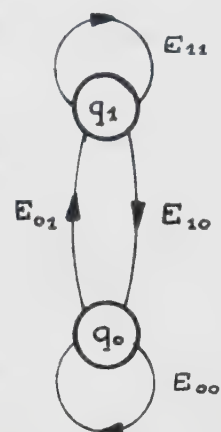
1°) Diagramme de fluence d'une telle machine.

L'ensemble des chemins possibles inclus dans l'ensemble des états d'entrées générables E_G est :

$$E_{11}, E_{10}, E_{01}, E_{00}.$$

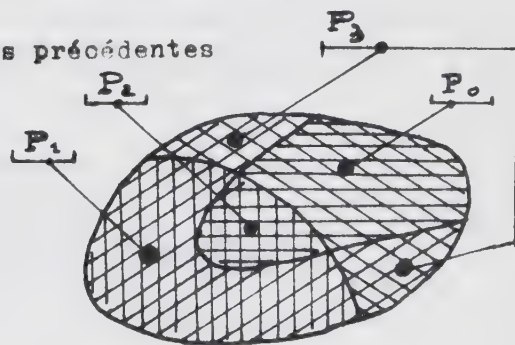
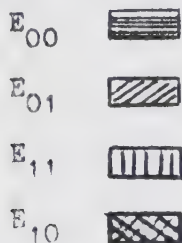
C'est une machine à évolutions consécutives dont les chemins doivent vérifier les propriétés suivantes :

$$\left[\begin{array}{l} E_{11} \cap E_{10} = \emptyset \text{ (théorème 8)} \\ E_{00} \cap E_{01} = \emptyset \text{ (théorème 8)} \\ N E_{10} \cap N E_{01} = \emptyset \text{ (théorème 11)} \\ E_{11} \cup E_{10} = E_G \text{ (théorème 8)} \\ E_{00} \cup E_{01} = E_G \text{ (théorème 8)} \end{array} \right.$$



2°) Recherche de la partition de E_G associée aux quatre sous-ensembles $E_{00}, E_{01}, E_{11}, E_{10}$, telle que toute partie de cette partition soit contenue dans au moins un de ces 4 sous-ensembles.

Représentons les propriétés précédentes



L'on détermine ainsi 4 parties P_0, P_1, P_2, P_3 deux à deux disjointes recouvrant E_G . *

* Nota : Ce résultat d'ailleurs usuel [15] montré par M. VALI et par nous indépendamment nous permet d'introduire l'équation générable de ces mémoires et sera étendu par la suite.

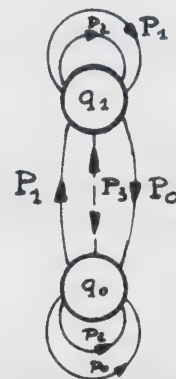
a) Une partie P_3 , dite de changement d'état de q
 $P_3 = E_{01} \cap E_{10} = (\varepsilon E_{01} \cup \delta E_{01}) \cap (\varepsilon E_{10} \cup \delta E_{10})$

b) Une partie P_2 dite d'état stable de q
 $P_2 = E_{00} \cap E_{11} = C_{EG} [E_{10} \cup E_{01}]$ (d'après Th 4 et Th 5)

c) Une partie P_1 , dite de mise à un de q
 $P_1 = E_{01} - E_3$

d) Une partie P_0 dite de mise à zéro de q
 $P_0 = E_{10} - E_3$

Le graphe au chemin peut se représenter de la manière classique suivante



3°) Equations aux partitions d'une telle machine

P_0, P_1, P_2, P_3 forment une partition de E_G .
 En fonctionnement normal (les entrées n'ont jamais d'état $\notin E_I$) nous pouvons appliquer la méthode classique utilisée pour les problèmes séquentiels particuliers, [toutes les variables ont l'état zéro, sauf une à la valeur 1].

$q(t)$	P_0	P_1	P_2	P_3
0	0	1	0	1
1	0	1	1	0

L'on peut effectuer la mise en équation de la variable interne q suivant plusieurs méthodes (associations des un et des zéro du tableau).

Recherchons les équations ne faisant pas apparaître la partie P_2 correspondant à l'état stable. Il vient :

$$\begin{aligned} q(t + \Delta t) &= \overline{P_0} (P_1 + q \overline{P_3} + \overline{q} P_3) \text{ à l'époque } t \\ \overline{q}(t + \Delta t) &= \overline{P_1} (P_0 + \overline{q} \overline{P_3} + q P_3) \text{ à l'époque } t \end{aligned}$$

que l'on peut écrire sous la forme classique :

$$\begin{aligned} s_q &= P_1 + \overline{q} P_3 \\ r_q &= P_0 + q P_3 \end{aligned}$$

L'on peut effectuer la mise en équations d'une telle machine uniquement en fonction de ces chemins non bouclés c'est-à-dire en fonction de E_{01} et E_{10} après avoir déterminé

$$\begin{cases} E_3 &= E_{10} \cap E_{01} \\ E_1 &= E_{01} - E_3 \\ E_0 &= E_{10} - E_3 \end{cases}$$

et reporté dans les équations explicites précédentes .

Nota : Nous remarquerons qu'en fonction du choix des évolutions de machine sur panne des entrées, l'on peut inclure ou non tout ou partie de E_I dans E_0 , E_1 , E_2 , E_3 .

4°) Application à l'étude des mémoires de la première espèce à niveaux.

Dans ces mémoires seuls les états niveaux des entrées sont actifs (impliquent des évolutions d'état interne en état interne)

$$E_{10} \cap E_{01} = \emptyset \text{ d'où } E_{10} = P_0 ; E_{01} = P_1$$

(la partition E_3 doit être vide, la fonction booléenne de $\overline{P_3} = 1$)

L'on pourra écrire directement l'équation de la mémoire en fonction des chemins non bouclés du diagramme de fluence c'est-à-dire soit la forme directe, soit la forme complémentaire

$$\begin{aligned} q(t + \Delta t) &= \overline{E_{10}} \cdot (E_{01} + q) \\ \overline{q}(t + \Delta t) &= \overline{E_{01}} \cdot (E_{10} + \overline{q}) \end{aligned}$$

Pour générer les chemins E_{00} , E_{10} , E_{01} , E_{11} , il faut au moins deux entrées :

$\{0, 1\}^2$, notons les s_e et r_e , les états des entrées possibles seront alors : e_0 , e_1 , e_2 , e_3

$\{E\}$	s_e	r_e
e_0	0	0
e_1	0	1
e_3	1	1
e_2	1	0

Discutons des propriétés de notre mémoire en fonction de l'appartenance ou non des états e_0, e_1, e_2, e_3 aux partitions P_0, P_1, P_2, P_3 .

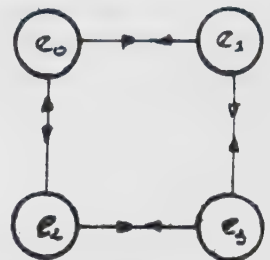
a) pour que ce soit une fonction mémoire il faut que P_2 ne soit pas vide.

b) Si un état appartient à P_3 il faut que ce soit un état interdit.

c) Classons ces mémoires en fonction du nombre d'états appartenant à P_0, P_1, P_2, P_3 (un état ne pouvant appartenir qu'à une seule partition).

Nous avons indiqué dans le tableau ci-dessous le nombre d'état appartenant à chaque partition.

Nous remarquerons que le graphe des évolutions possibles de s_e et r_e ci-contre implique que la partition P_2 est obligatoirement adjacente à P_1 et P_0



E_{10}	E_{01}	P_2	$\leq E_I$ P_3	Nom de la "mémoire"
P_0	P_1			
1	1	1	1	mémoire indéterminée "Vraie"
2	1	1		mémoire prioritaire à zéro
1	2	1		mémoire prioritaire à un
1	1	2		mémoire prioritaire stable
1		3		monostable à zéro
	1	3		monostable à un

L'on pourra aisément retrouver les équations des mémoires classiques en affectant e_0, e_1, e_2, e_3 à P_0, P_1, P_2, P_3 (P_3 étant interdit.)

Introduisons le signal forcé de mise aux conditions initiales par MZG

Il vient : si MZG effectue la mise à zéro

$$\begin{cases} q = \overline{\text{MZG}} \cdot \overline{E_{10}} (E_{01} + q) \\ \text{si MZG effectue la mise à un} \\ q = \overline{E_{10}} \cdot (E_{01} + q) + \text{MZG} \end{cases}$$

Name of Member		Address	
1	Mr. J. H. Smith	123 Main St.	Anytown, U.S.A.
2	Mr. J. H. Smith	123 Main St.	Anytown, U.S.A.
3	Mr. J. H. Smith	123 Main St.	Anytown, U.S.A.
4	Mr. J. H. Smith	123 Main St.	Anytown, U.S.A.
5	Mr. J. H. Smith	123 Main St.	Anytown, U.S.A.
6	Mr. J. H. Smith	123 Main St.	Anytown, U.S.A.
7	Mr. J. H. Smith	123 Main St.	Anytown, U.S.A.
8	Mr. J. H. Smith	123 Main St.	Anytown, U.S.A.
9	Mr. J. H. Smith	123 Main St.	Anytown, U.S.A.
10	Mr. J. H. Smith	123 Main St.	Anytown, U.S.A.

8. The eighth part of the document is a list of the names of the members of the committee who have been appointed to study the problem of the

9. The ninth part of the document is a list of the names of the members of the committee who have been appointed to study the problem of the

10. The tenth part of the document is a list of the names of the members of the committee who have been appointed to study the problem of the

11. The eleventh part of the document is a list of the names of the members of the committee who have been appointed to study the problem of the

12. The twelfth part of the document is a list of the names of the members of the committee who have been appointed to study the problem of the

13. The thirteenth part of the document is a list of the names of the members of the committee who have been appointed to study the problem of the

d) Application à la logique utilisée

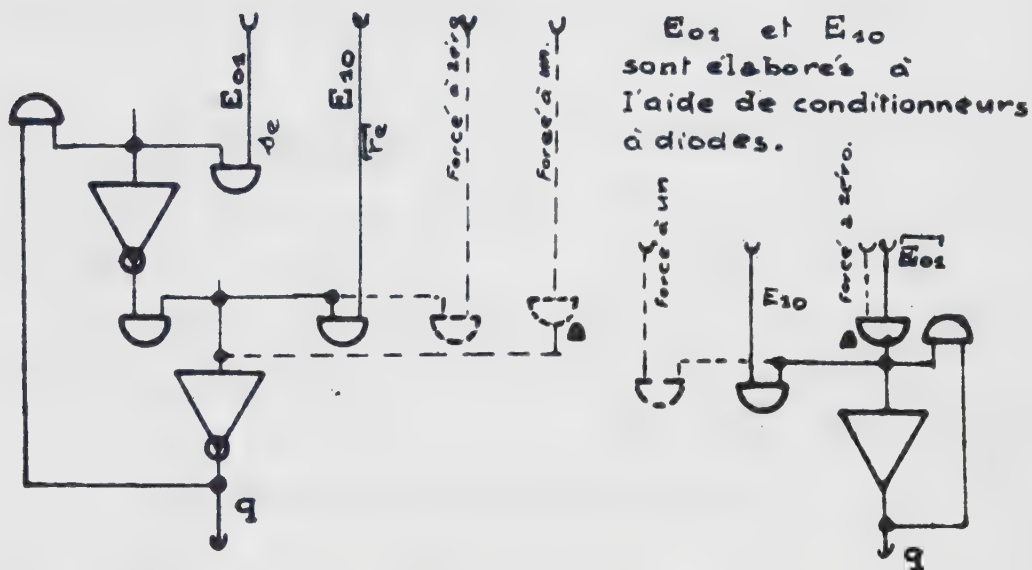
Précisions à titre d'exemple des mémoires du type mise à zéro prioritaire, telles que :

$$P_0 = e_1 + e_3, \quad P_1 = e_2, \quad P_2 = e_0.$$

L'équation de ces mémoires est :

$$q = \overline{r_e}(s_e + q)$$

mémoire qui se schématise de la manière ci-dessous à l'aide des circuits utilisés SDS, CII [16]



les circuits de mise à un, de mise à zéro forcés sont dessinés en pointillés.

5°) Remarques

• Les mémoires de la 1ère espèce, classiques ne comporte pas d'état $P_3 \notin E_I$

• Les mémoires du type RS indéterminé vrai ne sont pas employées ailleurs que dans les oscillateurs.

• Il reste trois types de mémoires :

- [prioritaire à zéro
- [prioritaire à un
- [prioritaire stable P Q

2-5-2 Etude des fonctions bascules

1°) Généralités

Considérons une machine de MOORE à deux variables internes $\{Q\} = \{q^1, q^2\}$ et un ensemble d'entrées $\{E\}$

• E_I est l'ensemble des états interdits de ces entrées

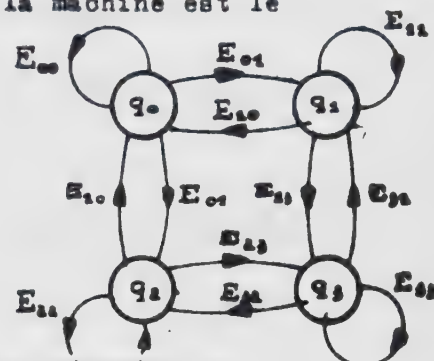
• $\{q_0, q_1, q_2, q_3\}$ sont les états niveaux des variables internes.

Cette machine sera telle que les évolutions d'état interne en état interne ne sont fonction que des états niveaux de $\{Q\}$ et de l'état des entrées à l'instant t .

Le diagramme de fluence de la machine est le suivant (voir ci-dessous)

Ces chemins, inclus dans E_G sont :

$$\begin{cases} E_{00}, E_{11}, E_{33}, E_{22} \\ E_{01}, E_{13}, E_{32}, E_{20} \\ E_{02}, E_{23}, E_{31}, E_{10} \end{cases}$$



Ils vérifient les propriétés suivantes :

$$\left. \begin{aligned} E_{ij} \cap E_{ik} &= \emptyset \\ N E_{ij} \cap N E_{ji} &= \emptyset \\ E_{ii} \cup E_{ij} \cup E_{ik} &= E_G \end{aligned} \right\} \begin{aligned} &\forall q_i \in Q \\ &q_j \text{ et } q_k \text{ étant les sommets} \\ &\text{consécutifs à } q_i. \end{aligned}$$

2°) Dénombrons les parties E_G associées aux 12 sous-ensembles E_{ij} ci-dessus

• étant un élément de E_G , l'équation de fonctionnement :

$$F[q_i(t), e(t)] = q_i(t + \Delta t) \quad \forall q_i \in Q_G$$

peut se traduire en disant que e définit une application f_e de Q_G dans Q_G .

L'ensemble des applications Q_G dans Q_G est notée $\mathcal{F}(Q_G, Q_G)$, il a n^n éléments, si n est le nombre d'éléments de Q_G .

Quand e décrit E_G l'application $e \xrightarrow{H} f(e)$ est une application de E_G dans $\mathcal{F}(Q_G, Q_G)$ notée H .

L'ensemble $\mathcal{F}(Q_G, Q_G)$ ayant n^n éléments nous noterons $f_i / i \in [1, n^n]$ ses éléments. On définit alors les n^n parties suivantes de E_G :

$$A_i = H^{-1}[f_i] \quad i \in [1, n^n]$$

[image réciproque de $\{f_i\}$ par l'application H , c'est-à-dire

$$A_i = \{e \in E_G \text{ tel que } H(e) = f_e = f_i\}$$

La définition des A_i montre que cette famille forme une partition de E_G .

Pour obtenir les A_i d'une machine à évolutions consécutives il faut éliminer tous les A_i tel que les applications f_i associent au moins deux états non consécutifs.

Dans le cas général il est très long d'énumérer tous les cas [4 états internes $\longrightarrow 4^4 = 256 A_i$]

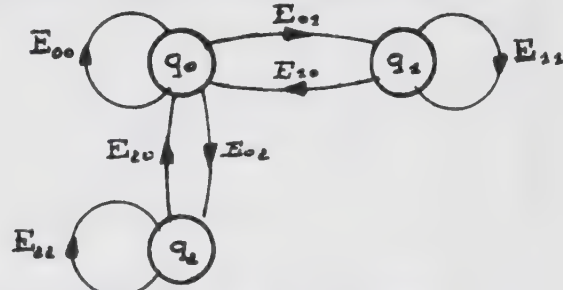
3°) Etudions une machine de MOORE ayant un état interdit

[par exemple q_3]

L'on remarquera que l'état interdit peut être quelconque, son choix ne change en rien la méthode d'étude.

Le diagramme de fluence est alors le suivant et ses chemins sont :

$$\begin{bmatrix} E_{00}, E_{11}, E_{22} \\ E_{01} \quad E_{02} \\ E_{10} \quad E_{20} \end{bmatrix}$$



a) Nous allons rechercher une partition dans l'ensemble des états E_G . Nous déterminerons le tableau des applications de $\{0, 1, 2\}$ dans $\{0, 1, 2\}$, soit $3^3 = 27$ applications. Après avoir éliminé les applications n'appartenant pas à la machine [telle que $q_2 \longrightarrow q_1$ ou $q_1 \longrightarrow q_2$], nous obtenons le tableau suivant, dans lequel nous avons dénommé symboliquement chaque application

$\{0,1,2\}$ $\rightarrow Q_G$	Partie A_1	$\{0,1,2\}$ $\rightarrow Q_G$	Partie A_1	$\{0,1,2\}$ $\rightarrow QG$	Partie A_1
012	C_2	112	P_1^1	212	P_1^2
010	P_0^2	110	B_0	210	P_3^2
~~011~~		~~111~~		~~211~~	
~~022~~		~~122~~		~~222~~	
002	P_0^1	102	P_3^1	202	B_1
~~020~~		~~120~~		~~220~~	
000	C_0	100	O_0	200	O_1
~~021~~		~~121~~		~~221~~	
~~001~~		~~101~~		~~201~~	

b) Tableau des évolutions aux partitions :

Nous avons repérer les états stables pour chaque partition par le signe ●

$\{Q\}$	P_0^1	P_1^1	P_3^1	P_0^2	P_1^2	P_3^2	C_0	C_2	B_0	B_1	O_0	O_1
0	●	1	1	●	2	2	●	●	1	2	1	2
1	0	●	0	●	●	●	0	●	●	0	0	0
3	INTERDIT											
2	●	●	●	0	●	0	0	●	0	●	0	0

q^1

c) Etablissons les équations de q_1 et de q_2 en appliquant la méthode classique (une partition à un, toutes les autres à zéro) afin de ne pas alourdir le tableau nous n'indiquerons $P_0, P_1, \delta, \epsilon$ juste pour les minterms où il y a une évolution.
(ils correspondent aux chiffres du tableau ci-dessus).

Pour q_2 nous avons :

$\{Q\}$	P_0^1	P_1^1	P_3^1	P_0^2	P_1^2	P_3^2	C_0	C_2	B_0	B_1	O_0	O_1
0		μ_0	μ_0		ε	ε			μ_0	ε	μ_0	ε
1	μ_0		μ_0				μ_0			μ_0	μ_0	μ_0
3												
2				δ		δ	δ		δ		δ	δ

q_2 q_1

$$q_2 = \overline{q_1} \cdot \left[P_0^2 + C_0 + B_0 + O_0 \right] \cdot \left[P_1^2 + B_1 + q_2 \left(P_3^2 + O_1 \right) + \overline{q_2} \left(P_3^2 + O_1 \right) \right]$$

Pour q_1 nous avons :

	P_0^1	P_1^1	P_3^1	P_0^2	P_1^2	P_3^2	C_0	C_2	B_0	B_1	O_0	O_1
0		ε	ε		μ_0	μ_0			ε	μ_0	ε	μ_0
2				μ_0		μ_0	μ_0		μ_0		μ_0	μ_0
3												
1	δ		δ				δ			δ	δ	δ

q_1 q_2

$$q_1 = \overline{q_2} \cdot \left[P_0^1 + C_0 + B_1 + O_1 \right] \cdot \left[P_1^2 + B_0 + q_1 \left(P_3^1 + O_0 \right) + \overline{q_1} \left(P_3^1 + O_0 \right) \right]$$

Table 1: Summary of Data			
Category	Sub-category	Value 1	Value 2
A	A1	10	20
	A2	15	25
	A3	20	30
B	B1	30	40
	B2	40	50
	B3	50	60
C	C1	60	70
	C2	70	80
	C3	80	90

Table 2: Detailed Data			
Category	Sub-category	Value 1	Value 2
A	A1	10	20
	A2	15	25
	A3	20	30
B	B1	30	40
	B2	40	50
	B3	50	60
C	C1	60	70
	C2	70	80
	C3	80	90

4°) Etablissons les chemins puis les équations aux chemins d'évolution:

$$\left[\begin{array}{l} E_{01} = P_1^1 \cup P_3^1 \cup B_0 \cup O_0 \\ E_{10} = P_0^1 \cup P_3^1 \cup B_1 \cup O_0 \cup O_1 \cup C_0 \\ E_{20} = P_0^2 \cup P_3^2 \cup B_0 \cup O_0 \cup O_1 \cup C_0 \\ E_{02} = P_1^2 \cup P_3^2 \cup B_1 \cup O_1 \\ E_{00} = P_0^1 \cup P_0^2 \cup C_0 \cup C_2 \\ E_{11} = P_1^1 \cup P_0^2 \cup P_1^2 \cup P_3^2 \cup C_2 \cup B_0 \\ E_{22} = P_0^1 \cup P_1^1 \cup P_3^1 \cup P_1^2 \cup C_2 \cup B_1 \end{array} \right.$$

Ils doivent vérifier les conditions aux chemins des machines déterministes.

$$\left[\begin{array}{ll} E_{01} \cap E_{02} = \emptyset & \longrightarrow \text{(vérifié)} \\ E_{01} \cap E_{10} = P_3^1 \cup O_0 & \longrightarrow \text{sont des transitions } \delta \text{ ou } \varepsilon \\ E_{02} \cap E_{20} = P_3^2 \cup O_1 & \longrightarrow \text{sont des transitions } \delta \text{ ou } \varepsilon \\ E_{10} \cup E_{11} = E_0 & \end{array} \right.$$

Les équations peuvent s'écrire en fonction des chemins :

$$\left[\begin{array}{l} q_2 = \overline{q_1} \cdot \overline{A_0^2} \cdot [A_1^2 + q_2 \cdot \overline{A_3^2} + \overline{q_2} \cdot A_3^2] \\ q_1 = \overline{q_2} \cdot \overline{A_0^1} \cdot [A_1^1 + q_1 \cdot \overline{A_3^1} + \overline{q_1} \cdot A_3^1] \end{array} \right.$$

avec :

$$\left[\begin{array}{l} A_0^2 = E_{20} - (E_{02} \cap E_{20}) \\ A_1^2 = E_{02} - (E_{02} \cap E_{20}) \\ A_3^2 = E_{02} \cap E_{20} \end{array} \right. \quad \text{(transitions)}$$

$$\left[\begin{array}{l} A_0^1 = E_{10} - (E_{01} \cap E_{10}) \\ A_1^1 = E_{01} - (E_{01} \cap E_{10}) \\ A_3^1 = E_{01} \cap E_{10} \end{array} \right. \quad \text{(transitions)}$$

L'on peut aussi écrire les équations de q_1 et de q_2 sous la forme classique :

$$\left[\begin{array}{l} q_2 = \overline{q_1} \cdot [P_0^2 + C_0 + B_0 + O_0 + q_2(P_3^2 + O_1)] \cdot [P_1^2 + B_1 + \overline{q_2}(P_3^2 + O_1) + q_2] \\ q_1 = \overline{q_2} \cdot [P_0^1 + C_0 + B_1 + O_1 + q_1(P_3^1 + O_0)] \cdot [P_1^1 + B_0 + \overline{q_1}(P_3^1 + O_0) + q_1] \end{array} \right.$$

D'où l'on tire les équations en s_q et r_q

$$\left[\begin{array}{l} s_{q_2} = P_1^2 + B_1 + \overline{q_2} \cdot (P_3^2 + O_1) \\ r_{q_2} = P_0^2 + B_0 + O_0 + q_2(P_3^2 + O_1) + q_1 \end{array} \right.$$

$$\left[\begin{array}{l} s_{q_1} = P_1^1 + B_0 + \overline{q_1}(P_3^1 + O_0) \\ r_{q_1} = P_0^1 + B_1 + C_0 + O_0 + q_1(P_3^1 + O_0) + q_2 \end{array} \right.$$

Soit en fonction des chemins d'évolutions appliqué 4-4

$$\left[\begin{array}{l} s_{q_2} = A_1^2 + \overline{q_2} \cdot A_3^2 \\ r_{q_2} = A_0^2 + q_2 \cdot A_3^2 + q_1 \\ s_{q_1} = A_1^1 + \overline{q_1} \cdot A_3^1 \\ r_{q_1} = A_0^1 + q_1 \cdot A_3^1 + q_2 \end{array} \right.$$

Nous avons ainsi les équations aux chemins de la machine de MOORE pour toutes ses évolutions déterministes. Nous avons établi la partition $\{P\}$ de l'ensemble des états générables maximum E_G de cette machine. Tout autre état conduisant à une évolution différente ne peut appartenir qu'à l'ensemble des états interdits d'entrées E_I .

c) Influence des états interdits des entrées.

Nous venons d'étudier les évolutions d'une machine de MOORE totalement déterminée. Nous avons obtenu toutes les évolutions déterministes possibles de cette machine en fonction des états générables possibles des entrées.

Considérons maintenant les états des entrées appartenant à l'ensemble des états interdits E_I , c'est-à-dire les états que les entrées ne doivent jamais atteindre [sauf par panne des entrées].

Ces états correspondront à des évolutions non déterministes de la machine. Nous n'allons pas effectuer une étude systématique de tous les cas mais indiquer les types d'évolutions possibles de la machine, celle-ci se trouvant dans un état $q_i \in \{Q\}$.

Nous avons, l'état q_3 étant toujours interdit, et $e_i \in E_I$

$$q_0, e_i \longrightarrow \{q_0, q_1, q_2\}$$

$$q_1, e_i \longrightarrow \{q_0, q_1\}$$

$$q_2, e_i \longrightarrow \{q_0, q_2\}$$

Deux types d'évolutions sont possibles :

- les évolutions déterminées telles que

$$q_j, e_i \longrightarrow q_k \quad q_j \text{ et } q_k \text{ étant connu.}$$

Ce sont toutes des évolutions que nous avons étudiées précédemment, dans ce cas e_i est à inclure dans les partitions précédentes $\{P\}$ établies sur E_G .

- les évolutions indéterminées telles que :

$$q_j, e_i \longrightarrow q_k \quad \text{inconnu}$$

C'est-à-dire par exemple $q_0, e_i \longrightarrow q_1$ ou q_2 de façon aléatoire.

e_i ne pourra appartenir qu'à l'ensemble des états interdits E_I de la machine.

Evolutions indéterminées possibles

Afin de simplifier nous noterons les états indéterminés d'arrivée de la machine classique par exemple (1,2).

[Partant de q_0 nous avons (0, 1), (0, 2), (1, 2), (0, 1, 2)
 Partant de q_1 nous avons (0, 1)
 Partant de q_2 nous avons (0, 2)

Pratiquement seule l'évolution $q_0 \longrightarrow (1, 2)$ est intéressante, les autres étant stables de façon aléatoire. Déterminons la table des évolutions possibles à partir de $q_0 \longrightarrow (1, 2)$.

	I_0	I_1	I_2	I_3
0	1, 2	1, 2	1, 2	1, 2
1	•	0	•	•
3				
2	•	•	•	•

$\begin{array}{|l} \text{I} \\ \text{I} \end{array}$
 $\begin{array}{|l} \text{I} \\ \text{I} \end{array}$
 $\begin{array}{|l} \text{I} \\ \text{I} \end{array}$

$q_2 \quad q_1$

Nous avons alors les équations générales de cette machine en incluant les partitions des états interdits I_0, I_1, I_2, I_3 dues aux évolutions non déterministes

$$q_0 \xrightarrow{1} (1, 2) .$$

Ecrivons uniquement les équations sous la forme suivante :

$$\begin{cases} s_{q_0} = [P_1^2 + B_1 + I_1 + I_2] + \overline{q_2}(P_3^2 + O_1 + I_0 + I_3) \\ r_{q_0} = [P_0^2 + C_0 + B_0 + O_0] + q_2(P_3^2 + O_1 + I_0 + I_3) + q_1 \\ s_{q_1} = [P_1^1 + B_0 + I_0 + I_2] + \overline{q_1}(P_3^1 + C_0 + I_1 + I_3) \\ r_{q_1} = [P_0^1 + B_1 + C_0 + O_0] + q_1(P_3^1 + C_0 + I_1 + I_3) + q_2 \end{cases}$$

d) Conclusions :

Il est bien évident que les équations que nous obtenons sont les équations de deux bascules à impulsions dans lequel l'on aurait interdit l'état $q_1 = q_2 = 1$.

Nous avons étudié ce cas car il permet d'établir les équations de fonctionnement possibles d'une bascule.

L'étude avec les 4 états $\{q_0, q_1, q_2, q_3\}$ est plus lourde et ne conduit pas à des résultats à notre avis plus intéressants.

3°) Applications aux fonctions bascules à niveau.

Nous n'étudierons pas un opérateur de logique capable d'effectuer toutes les évolutions que nous venons de déterminer, mais seulement une partie.

Nous allons simplement rappeler quelques notions de base. Puis nous préciserons les caractéristiques des bascules utilisées pour notre machine.

a) Généralités relatives aux bascules à niveau.

Ce sont des machines de MOORE à deux sorties notées parfois F et $\overline{F}$. Nous noterons F_1 pour $\overline{F}$ et F_2 pour F .

Reprenons la machine de MOORE que nous avons établie précédemment et précisons les parties de la partition qui doivent appartenir à notre opérateur pour qu'il fonctionne comme une bascule $[F_1 = \overline{F_2}$ époques de transitions exclues].

① Aucune partition P_3^1, P_3^2, C_0, O_1 ne peut appartenir à l'opérateur [ce ne sont pas des niveaux].

② Tous les états des entrées conduisant à l'état stable q_0 c'est-à-dire les parties P_0^1, P_0^2, C_0 ne peuvent appartenir qu'à l'ensemble des états interdits des entrées de la bascule.

③ Ce doit être une bascule, B_0 et B_1 appartiennent aux états générables des entrées de l'opérateur.

④ Elle doit être stable au moins aux deux états q_1 et q_2 , c'est-à-dire qu'au moins une partie

P_1^1, P_1^2, C_2, I_2 appartient à l'ensemble des entrées générables. [I_2 peut être inclu dans le sous-ensemble des parties stables car aucune partie appartenant aux états générables de la bascule ne peut commander une évolution vers q_0 stable.]

Résumons cela sur le tableau des évolutions aux partitions.

	bascule		stable				Interdit		
	B_0	B_1	P_1^1	P_1^2	C_2	I_2	P_0^1	P_0^2	C_0
0	1	2	1	2	•	1,2	•	•	•
1	•	0	•	•	•	•	0	•	0
3	•	•	•	•	•	•	•	•	•
2	0	•	•	•	•	•	•	0	0

$F_2 \quad F_1$
 $F \quad \overline{F}$

Notons : B_2 l'ensemble des parties stables possibles

$$B_2 = P_1^1 \cup P_1^2 \cup C_2 \cup I_2$$

Comme pour les fonctions mémoires à niveau nous retrouvons les trois sous-ensembles caractéristiques B_0, B_1, B_2

formant une partition sur E_G . Mais l'on fera attention à

$B_I = P_0^1 \cup P_0^2 \cup C_0$; tout état des entrées appartenant à B_I sera forcément un état interdit. B_I sera dit ensemble des états de contraintes ; en fonctionnement bascule les états des entrées doivent être tels que $B_I = 0$.

b) Bascules à niveau simples

Il faut au moins trois parties appartenant à E_G pour que l'on ait une bascule. Cela nécessite au minimum 2 entrées s_e et x_e d'états $E = \{e_0, e_1, e_2, e_3\}$

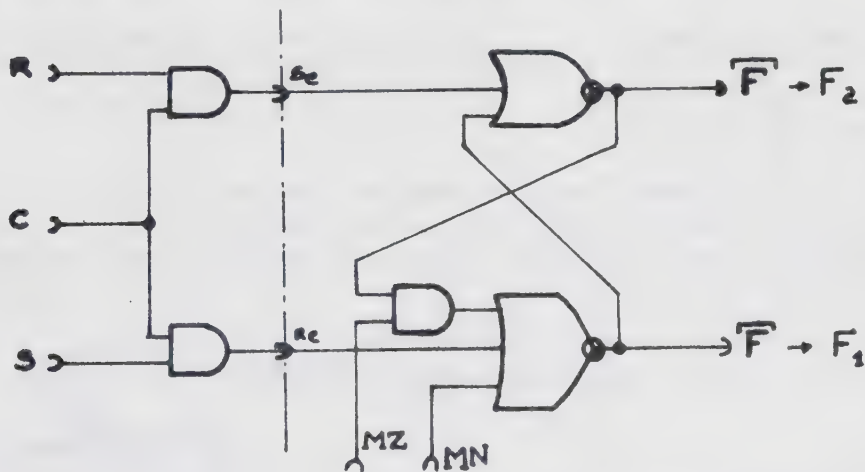
En fonction de l'affectation de B_0, B_1 , de la partie stable choisie et d'une autre partie choisie parmi toutes les parties ci-dessus, l'on détermine les bascules simples à deux entrées.

Rappelons que comme pour les mémoires il faut que B_0 et B_1 aient des états consécutifs à une partie stable.

c) Exemple de bascule à niveau utilisée [type RSC]

Ce sont des bascules à 4 entrées [si l'on exclut les circuits auxiliaires c'est-à-dire les portes d'intersection avec un ou plusieurs signaux d'horloge].

Nous allons étudier la bascule dont le schéma de base est le suivant (nous avons utilisé la symbolisation prévue par la normalisation).



Les équations sont classiques en fonction de s_e, x_e, MZ, MN nous avons :

$$F_1 = \overline{F_1} \cdot \overline{r_e}$$

$$F_1 = \overline{F_2} \cdot \overline{MZ} \cdot \overline{s_e} \cdot \overline{MN}$$

$$F_2 = \overline{s_e} \cdot (s_e + MN + F_2 \cdot MZ)$$

B_0 et B_1 sont des états d'évolutions bascules

• Nous avons deux fonctionnements de base pour cette bascule, ils se déterminent facilement sur le tableau ci-dessus :

- Le fonctionnement en bascule classique type RS obtenu par des séquences de s_e et x_e , $\overline{MN} \cdot \overline{MZ} = 1$.

l'on remarquera que la bascule est indéterminée en fonctionnement RSC si $R.S.C = 1$

- Le fonctionnement en initialisation par MN et MZ si $\overline{s_e} \cdot \overline{x_e} = 1$. L'opérateur se comporte aussi comme une bascule mais avec aléas éventuels en l'état q_3 . Nous dirons que c'est une bascule prioritaire à la mise à un (état q_0)

• L'on peut aussi décrire des séquences mixtes (à l'aide de s_e , x_e , MN , MZ) que l'on établira facilement sur le tableau ci-dessus.

• Ecrivons les équations de fonctionnement de notre opérateur.

En fonctionnement combinatoire [cas d'une bascule non utilisée sur une carte ou dans un "chips"], MN et MZ étant programmés à l'un des deux états ci-dessous, l'on a :

$$\begin{aligned} \text{si } \overline{MN} \cdot \overline{MZ} = 1 & \quad F_1 = \overline{s_e} ; \quad F_2 = \overline{x_e} \cdot s_e \\ \text{si } MN = 1 & \quad F_2 = \overline{x_e} \end{aligned}$$

En fonctionnement bascule nous aurons, en reprenant les notations classiques, les 3 équations suivantes :

$$\begin{cases} s_e \cdot x_e + MN \cdot x_e = 0 & \text{contrainte.} \\ sF = s_e + MN & = B_1 \\ rF = x_e + \overline{MN} \cdot \overline{MZ} \cdot \overline{s_e} & = B_0 \end{cases}$$

que nous nous permettrons d'écrire de la façon suivante pour les fonctionnements classiques où l'on respecte

$$\begin{aligned} \overline{s_e} \cdot \overline{x_e} = 1 & \quad \text{si } MN \text{ et } MZ \text{ décrivent une séquence} \\ \overline{MN} \cdot Z = 1 & \quad \text{si } x_e \text{ et } s_e \text{ décrivent une séquence.} \end{aligned}$$

$$\begin{cases} s_e \cdot x_e + MN \cdot x_e = 0 \\ sF = s_e + MN \\ rF = x_e + \overline{MZ} \end{cases}$$

3 - GENERALITES SUR LA COMMANDE NUMERIQUE ETUDIEE

pages :

3-1- <u>Les familles types de machines-outils à commande numérique.</u>	52
3-2- <u>Structure globale de la commande numérique étudiée.</u>	53
3-2-1- Structure de base	53
3-2-2- Structure globale	54
3-3- <u>Les ordres manuels.</u>	55
3-3-1- La programmation manuelle de démarrage	56
3-3-2- Les états globaux de la machine	56
3-3-3- La programmation de la mise en place de la bande	57
3-3-4- La programmation en usinage automatique	57
3-3-5- La programmation en usinage semi-automatique.	58
3-4- <u>Notions de gamme d'usinage et de programmation.</u>	58

3-1- Les familles types de machines-outil à commande numérique :

Les commandes numériques des machines-outils peuvent se classer, d'après le mode de programmation des trajectoires à usiner, en deux grandes familles :

- d'une part les machines à "programmation" de trajectoires élémentaires.
- d'autre part les machines à "programmation" de trajectoires incrémentales.

3-1-1- Les machines à programmation de trajectoires dites élémentaires :

Les trajectoires élémentaires sont en général des segments de droites, des arcs de circonférence, des arcs de paraboles...

La génération des trajectoires est faite par un calculateur d'interpolation interne spécialisé dit interpolateur. Les paramètres déterminant ces trajectoires : les coordonnées des points extrêmes, les coordonnées des centres d'arcs de circonférences, les rayons, etc..., sont calculés manuellement ou à l'aide d'un ordinateur, par un préparateur d'usinage et transmis à la commande numérique par l'intermédiaire d'une bande programme codée.

3-1-2- Les machines à "programmation" de trajectoire dites incrémentales :

Les trajectoires à usiner sont générées de point en point très voisins (un incrément des systèmes de mesure des déplacements).

Le calcul de ces points est effectué par un ordinateur qui peut être couplé soit :

a) directement à la commande numérique, en temps réel. La commande de la machine-outil se résume alors à un interface ordinateur, machine-outil.

b) indirectement à la commande numérique par l'intermédiaire d'une bande magnétique élaborée par le calculateur et utilisée, lorsque l'opérateur le souhaite, sur la machine-outil.

.../...

3-2- Structure globale de la commande numérique étudiée

Nous avons étudié une commande numérique de machine-outil de la première famille, c'est à dire une commande :

- programmée, en usinage automatique par les données d'une bande programme codée , en usinage semi-automatique par affichage direct alphanumérique.
- pilotée en fonction des programmes et de l'état de la machine par un opérateur (dit aussi servant de la M.O.), agissant sur un pupitre.
- générant les trajectoires à usiner par un calculateur interne d'interpolation de trajectoires (arcs de cercles et segments de droites).

3-2-1- Structure logique de base :

Une machine-outil à commande numérique est un système dont la structure de base peut être définie comme pour tous les systèmes, de deux manières différentes :

- par sa structure logique.
- par sa structure technologique.

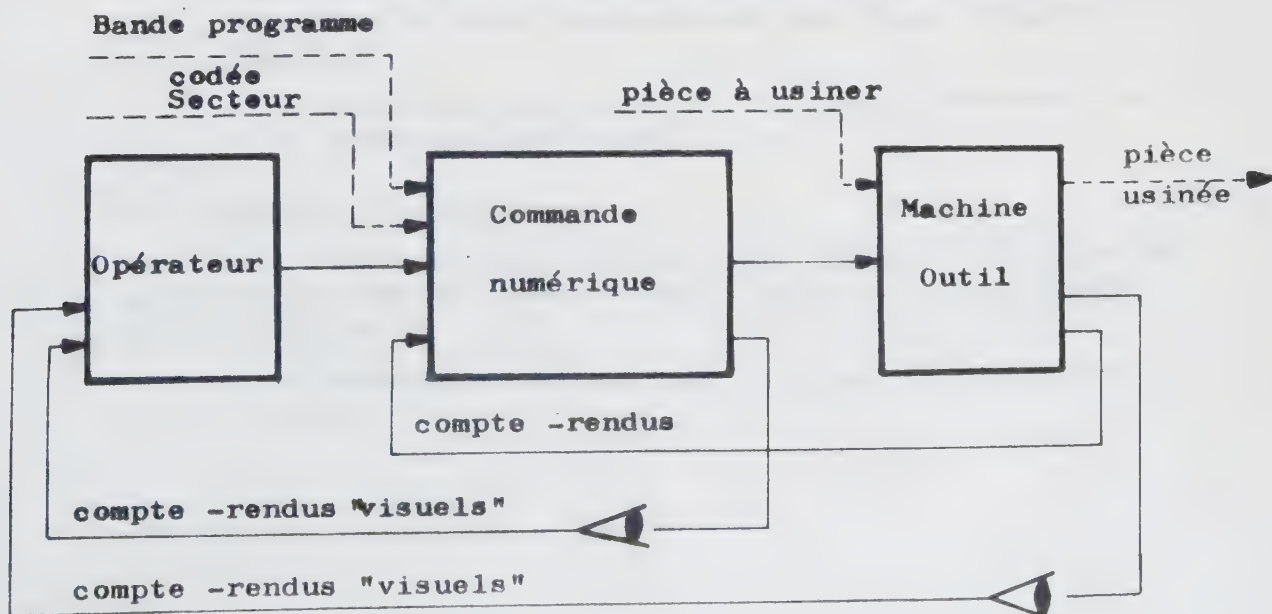
Nous nous limiterons à la structure logique, bien que, lors de la conception d'un système il soit nécessaire de développer simultanément les deux structures.

Une machine-outil à commande numérique et son opérateur forment une machine autonome (il n'en sort, ni n'entre aucune information logique lorsque ce dernier a choisi une bande programme codée, à usiner).

Elle sera composée de trois parties :

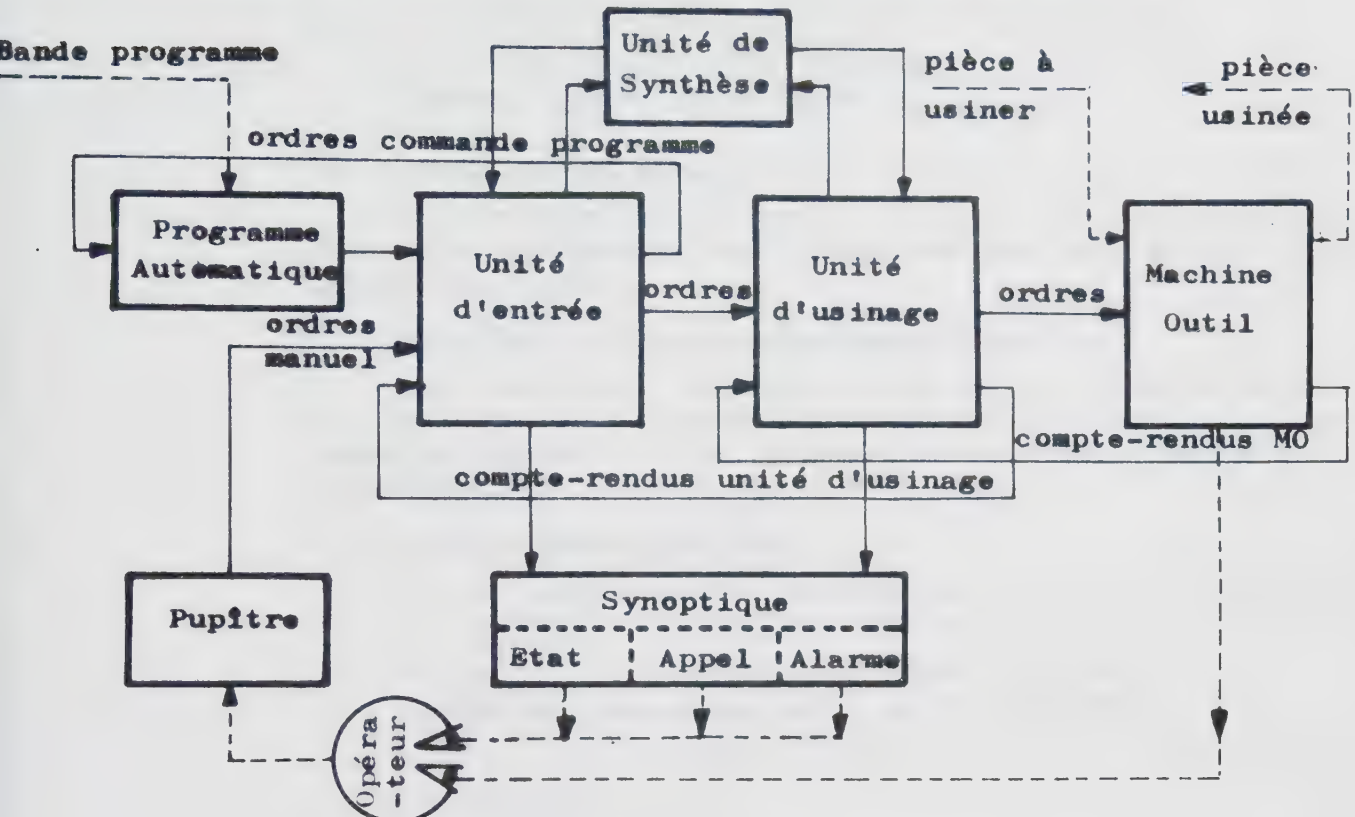
- la machine-outil
- la commande numérique
- l'opérateur

Nous pouvons la représenter par le schéma ci-après.



3-2-2- Structure globale :

La commande numérique étudiée se décompose en unités logiques fonctionnelles suivant le schéma ci-dessous :



Nous avons fait apparaître :

1°) L'unité d'entrée dont les fonctions sont :

- la lecture et la vérification syntaxique des informations de la bande programme codée puis la répartition des informations lues dans l'unité d'usinage.
- la répartition des informations affichées manuellement en usinage semi-automatique dans l'unité d'usinage.
- la programmation manuelle des états fondamentaux de la commande numérique.

2°) L'unité d'usinage :

Elle commande toute la machine-outil en fonction des informations émises par l'unité d'entrée. Elle commande en particulier :

- les déplacements des axes (trajectoires et points d'arrêts)

.../...

- la tête d'usinage de la machine-outil (vitesse d'avance, vitesse de rotation de la broche, cycles,...)
- le choix des outils
- les opérations annexes (serrage pièce...)
- la mise en état des machines périphériques (centrale hydraulique, alimentations, ...).

3°) L'unité de synthèse :

En fonction des états atteints par l'unité d'entrée et l'unité d'usinage, elle commande les évolutions suivantes de ces unités.

4°) Le lecteur de bande programme codée :

Sur ordre de l'unité d'entrée il lit (mécaniquement par position de contact par exemple) un caractère de la bande programme et fait avancer ou reculer la bande de un caractère.

5°) Le synoptique regroupe :

- les compte-rendus visuels des états fondamentaux de la machine et de sa commande.
- les appels visuels indiquent à l'opérateur les commandes manuelles qu'il peut effectuer.
- les compte-rendus visuels de défaut et d'alarme du système.

6°) Le pupitre regroupe :

- les boutons poussoirs de la programmation manuelle des états globaux de la machine.
- les affichages des informations manuelles en usinage semi-automatique.
- les commandes de mise en service du système et les commandes manuelles d'alarmes.

3-3- Les ordres manuels :

Dans ce paragraphe nous citerons les ordres de programmation manuelle (groupés sur le pupitre de la commande numérique de la machine-outil).

Nous avons essentiellement déduit ces ordres d'une étude type organisation du travail qui n'a pas sa place dans cette thèse (nous pourrions les considérer comme des données). Les ordres de programmation manuelle dit d'option de sous-programme, bien que cités

.../...

ici, sont en fait déduits de l'étude des ordinogrammes de lecture de bandes programme codées (chapitre 5).

L'opérateur dispose d'un pupitre qui lui permet :

1°) D'une part d'intervenir aux moments opportuns dans le déroulement des opérations de la machine-outil, en fonction :

- a) des impératifs de mise en conditions initiales du système c'est à dire : du démarrage de la machine (mise sous tension...), de la mise en place de la bande programme codée, et du calage à zéro des différents axes de la machine.
- b) des impératifs et des options du programme codé machine-outil, apparaissant lors de son déroulement.
- c) des anomalies de fonctionnement, non détectées par la logique, constatées visuellement par l'opérateur dans le système.

2°) D'autre part, de connaître à chaque instant, l'état des fonctions essentielles de la machine.

3-3-1- La programmation manuelle de démarrage de la machine :

Ce sont les fonctions de mise sous-tension et d'arrêt général de la commande de la machine.

Nous remarquerons que lors de la mise sous tension d'un système logique, les mémoires et les bascules prennent, à priori, un état quelconque. Nous effectuerons, à la mise sous tension, une mise aux conditions initiales, forcée, automatique.

3-3-2- La programmation manuelle des états globaux de la commande :

La commande a trois états possibles, commandés par les ordres manuels suivants :

- a) La Préparation du Lecteur : symbole PL

Permet de mettre en place et aux conditions initiales la bande programme codée.

- b) L'Usinage Automatique : symbole UA

Dans cet état la bande programme codée commande la machine, toutes les actions de l'opérateur ne sont efficaces que si elles sont compatibles avec les informations que comporte le programme codé.

.../...

c) L'Usinage semi-automatique; symbole U S A

Dans cet état, la machine-outil évolue uniquement en fonction du programme manuel affiché par l'opérateur sur le pupitre et l'on ne peut pas mettre en place la bande.

3-3-3- La programmation manuelle de mise en place de la bande :

Elle permet de placer la bande sur le lecteur en une position telle que l'on soit au début d'un programme codé et d'enrouler ou de dérouler la bande. Cinq ordres sont à la disposition de l'opérateur :

- mise en place de la bande : symbole MPB
- de caractère à caractère : symbole Pà P
- de programme à programme : symbole Pi à Pi
- dans le sens avance bande : symbole Av
- dans le sens retour bande : symbole Re.

Lorsque la bande a atteint un caractère de début de programme, l'opérateur contrôle visuellement si c'est le début du programme cherché, et en informe la machine à l'aide de l'ordre manuel ; Bande prête : symbole BP.

En mise en place de la bande, l'opérateur peut arrêter le déroulement en agissant à n'importe quel instant sur Stop Bande : symbole Stop B.

3-3-4- La programmation manuelle en usinage automatique :

L'opérateur programme manuellement le mode de lecture de la bande programme codée c'est à dire, soit la Lecture Manuelle (symbole L M) de Caractère à Caractère (symbole C à C) ou de Bloc à Bloc (symbole B à B), soit la Lecture Automatique (symbole L A).

Il peut démarrer et arrêter manuellement la lecture et l'usinage d'un bloc à l'aide des ordres suivants :

- . Impulsion d'usinage et de lecture : symbole Imp U
- . Stop lecture et usinage : symbole Stop M
- . Arrêt d'Urgence de la lecture et de l'usinage : symbole Ar Ug

. Choisir manuellement les options possibles dans les cas des sous-programmes répétitifs ou optionnels c'est à dire soit répéter un sous programme répétitif (symbole REP), soit passer un bloc Optionnel (symbole Opt), soit Annuler les options de sous-programme ci-dessus (symbole Ann).

.../...

3-3-5- La programmation manuelle en usinage semi-automatique :

L'opérateur peut mettre en mémoire, l'un après l'autre, dans un ordre quelconque, les mots nécessaires pour effectuer un usinage déterminé (uniquement en point à point). Pour ce faire, il dispose de rotacteurs d'affichage et d'un ordre de mise en mémoire par registre (symbole Mn N). Nous ne détaillerons pas plus dans le cadre de notre Thèse, car les informations ainsi programmées ne sont pas traitées par l'unité d'entrée de la machine-outil, mais directement par l'unité d'usinage.

L'opérateur peut déclencher et arrêter manuellement l'usinage du bloc pré-affiché à l'aide des mêmes ordres de démarrage et d'arrêt, que dans le cas de l'usinage automatique.

3-4- Notions de gamme d'usinage et de programmation :

Etant donnée une pièce à usiner sur une machine outil à commande numérique déterminée, l'on dispose :

- Des données pièce, c'est à dire :

- . du plan de la pièce, usinages terminés.
- . du plan ou des caractéristiques (imprécises) de la pièce avant usinage.
- . des données technologiques de la pièce à usiner: (matériaux, résistances, vibrations pièce, déformations par libération des contraintes internes...).

- Des données machine-outil, c'est à dire :

- . des caractéristiques de la machine proprement dite et en particulier ses caractéristiques limites (précision, courses, nombre d'axes, gamme des vitesses, puissances...)
- . des caractéristiques de la commande numérique (programme automatique, programme semi-automatique...).

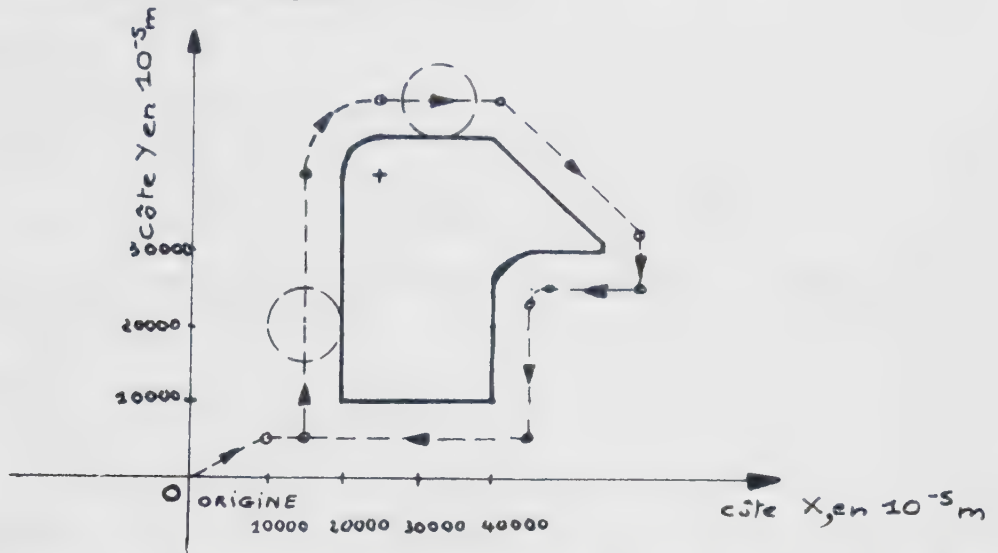
- D'une collection d'outils disponibles.

Partant du plan de la pièce terminée, le problème est de déterminer une bande programme codée qui commandera toutes les évolutions de la machine-outil afin de réaliser la pièce en respectant toutes les contraintes (toutes les autres données).

Deux méthodes permettent d'établir cette bande programme, l'une que l'on appelle programmation manuelle, l'autre que l'on dénomme programmation assistée par un ordinateur.

.../...

Brièvement, à l'aide d'un exemple simple, nous allons décrire les opérations que doit effectuer une machine-outil à commande numérique afin d'usiner la pièce représentée ci-dessous. Nous nous limiterons à la méthode de programmation manuelle, sans tenir compte des calculs et précautions classiques.



La programmation manuelle est effectuée directement par un préparateur (en usinage) qui décrit en français les opérations à effectuer afin de réaliser la pièce donnée, sur la machine-outil choisie.

Le préparateur établit la gamme d'usinage (très simplifiée) suivante :

Opération préparatoire 1 : (en semi-automatique)

- mettre la pièce en place sur la machine
- serrer la pièce
- monter l'outil numéro 2 (lunette de calage)
- caler l'origine de la pièce par rapport à la machine.

Opération préparatoire 2 :

- mettre en place la bande programme

Opération (1) :

- en point à point, aller en $X = + 10000$ $Y = + 5000$
- à la vitesse d'avance de 1800 mm/mn
- monter l'outil n°3 (fraise $\phi 500$; 6 dents)

Opération (2) :

- En contournage linéaire aller en $X = + 15000$ $Y = + 5000$
- à la vitesse d'avance de 300 mm/mn
- la broche tournant à la vitesse de 400 tr/mn
- en lubrifiant.

.../...

Opération (3) :

- en contournage linéaire aller en $X = + 3000$ $Y = + 4000$
- les autres conditions précédentes, ci-dessus, étant identiques.

Opération (4) :

- en contournage circulaire de sens horaire
- usiner l'arc de circonférence

d'extrémité $X = + 25000$ $Y = + 50000$
 de centre $I = + 25000$ $J = + 40000$

- les autres conditions étant identiques à celles de l'opération précédente.

Opération (5) :

⋮

Opération (N)

- dégager l'outil de la pièce vers $X = + 0$ $Y = + 0$
- à la vitesse d'avance rapide
-

Opération N + 1

- rester en place $X = + 0$ $Y = + 0$
- arrêter la broche
- démonter l'outil n°3
- desserrer la pièce
- l'usinage de la pièce est terminé.

Cette gamme d'usinage, écrite en français, est incompréhensible pour une machine classique actuelle. Il est nécessaire d'introduire un langage spécial dit machine-outil qui décrit de façon claire, simple, de manière biunivoque, toutes les actions élémentaires (Ex. en contournage circulaire de sens horaire, usiner l'arc de circonférence) permettant d'effectuer une opération puis de proche en proche toutes les opérations qui permettent d'usiner la pièce.

4 - SYNTAXIQUE D'UN LANGAGE MACHINE-OUTIL

pages :

4-1- <u>Méthode dite de Naur et de Backus - Rappels des notions utilisées pour l'étude.</u>	63
4-2- <u>L'Alphabet</u>	65
4-2-1- Alphabet machine-outil	65
4-2-2- Alphabet de l'imprimante	66
4-2-3- Alphabet codé	66
4-3- <u>Les mots</u>	67
4-3-1- Les mots HORS programme	67
4-3-2- Les mots EN programme	67
4-3-3- Mots en programmes remarquables	68
4-4- <u>Les blocs</u>	69
4-4-1- Définitions	69
4-4-2- Types de Blocs	69
4-4-3- Syntaxique des blocs optionnels	70
4-4-4- Blocs remarquables.	70
4-5- <u>Les programmes</u>	71
4-5-1- Notions	71
4-5-2- Types de programme	72
4-5-3- Sous-programmes remarquables	73
4-5-4- Rupture de séquence programme	75
4-6- <u>Les bandes programmes codées</u>	75
4-6-1- Rappels, Alphabet codé machine à écrire	75
4-6-2- Programme codé	76
4-6-3- Bande programme codée	76
4-7- <u>Conclusions</u>	76
4-7-1- Remarques relatives aux langages ISO et EIA	76
4-7-2- Conclusions.	77

Introduction.

La bande programme codée Machine-outil comporte toutes les données suffisantes permettant d'effectuer l'usinage d'une pièce souhaitée et ce, quel que soit l'usinage, à condition qu'il soit compatible avec les caractéristiques limites de la machine.

Cette bande devra être "écrite" en un langage codé compréhensible de façon biunivoque par la machine.

Le langage machine-outil est un langage informationnel qui s'articulera comme un langage parlé, en :

- programme : ensemble des données permettant de réaliser une pièce sur la machine,
- formé par des blocs : ensemble des données permettant d'effectuer des opérations (ou trajectoires élémentaires),
- formés par des mots : ensemble des données permettant d'effectuer des actions élémentaires,
- formés par des symboles (ou caractères) appartenant à un alphabet.

A chaque mot, bloc, programme, du langage machine-outil, correspondra par l'application Sémantique, une action élémentaire, une opération, une pièce particulière, décrite habituellement par un plan et une gamme d'usinage rédigées en langage parlé. [Voir paragraphe 3-4].

Notre but ici, n'est pas de rechercher un nouveau langage, il en existe déjà un assez grand nombre, dont un, normalisé par le comité de Normalisation française qui nous servira d'exemple d'application.

Nous nous proposons, dans ce chapitre, d'étudier théoriquement à l'aide des Formes de Naur et de Backus, la syntaxique d'un langage général machine-outil, c'est à dire, l'ensemble des règles "grammaticales" d'écriture de ce langage. Pour ce faire, nous reconstruirons le langage à partir de l'Alphabet jusqu'au programme.

Nous ne développerons pas les assignations informationnelles (ou application sémantique de ce langage), nous renverrons le lecteur aux planches jointes.

Nous ne nous occuperons pas de la syntaxique d'usinabilité des données précisées sur la bande programme, cette syntaxique étant vérifiée par l'unité d'usinage. (Exemple : si l'on veut effectuer un usinage de contourage sur une pièce, que la pièce n'est pas bridée sur la table de la machine, il y a alors une erreur de syntaxique d'usinabilité qui sera détectée par l'unité d'usinage).

4 -1- Méthode dite de Naur et Backus - Rappels des notions utilisées pour l'étude.

4 -1-1- Généralités :

L'étude théorique des règles grammaticales d'écriture (dites syntaxiques) d'un langage informationnel, peut être conduite suivant différentes méthodes plus ou moins lourdes et denses.

Deux méthodes, en particulier, sont possibles :

- La méthode d'étude que nous appellerons ensembliste (théorie des ensembles avec son vocabulaire, sa syntaxe, ses théorèmes et définitions).
- La méthode dite de Naur et de J.N. Backus. C'est l'écriture condensée synthétique des propriétés syntaxiques d'un langage informationnel à l'aide de métalangue de J.N. Backus.

Nous allons préciser cette métalangue en nous limitant aux propriétés utilisées dans notre étude.

Puis, pour un exemple choisi dans les paragraphes suivants, nous écrirons les propriétés syntaxiques à l'aide :

- de la métalangue de Backus,
- du langage "ensembliste".

Le lecteur vérifiera la richesse et la densité des informations que comportent les règles syntaxiques écrites à l'aide de la métalangue.

4 -1-2- La méthode de Naur et de Backus :

Elle se compose d'un alphabet, de règles syntaxiques d'écriture et de lecture (extraites de l'ouvrage [1] que nous nous sommes permis d'alourdir, afin de simplifier l'étude syntaxique du langage machine-outil.

1°) L'Alphabet :

::= sens français : définition. "La variable métalinguistique située à la gauche de ::=, est définie par les variables métalinguistiques ou les variables du langage machine-outil, ou encore leurs combinaisons, situées à droite". [1]

| sens français : OU (dit exclusif). "Si une variable métalinguistique peut prendre plusieurs valeurs, celles-ci sont séparées dans la définition par le symbole de concaténation : | ". [1]

.../...

- ④ avec : $\langle \text{mots punctuation} \rangle ::= \langle \text{Fin de Bloc} \rangle | / | \langle \text{Blanc} \rangle$
 ⑤ avec : $\left[\begin{array}{l} \langle \text{Fin de Bloc} \rangle ::= \text{FB} | \text{FB} \langle \text{Fin de bloc} \rangle \\ \langle \text{Blanc} \rangle ::= \text{BL} | \text{BL} \langle \text{Blanc} \rangle \end{array} \right.$
 ⑥ etc...

2°) Rappel de terminologie :

- , $\langle \text{mots} \rangle$ est une variable métalinguistique
 / est une variable langage de la machine outil.

3°) Définitions et propriétés dans le langage "ensembliste" :

Définition ① : L'ensemble des mots est la réunion des mots EN programme et HORS programme.

Propriété : L'intersection des mots en programme et hors programme est vide.

Définition ② : L'ensemble des mots Hors programme est la réunion de :

- (i) un caractère a_i compris entre les parenthèses (et), dans cet ordre :

$$a_i \in \mathcal{A}_{MI}; \text{ (et) exclus.}$$

- (ii) un sous ensemble de caractères, non ordonné fini $a_i, a_j, \dots$, compris entre les parenthèses (et) dans cet ordre, $a_i, a_j \in \mathcal{A}_{MI}; \text{ (et) exclus.}$

Définition ③ : L'ensemble des mots EN programme est la réunion de l'ensemble des mots punctuation, des mots adresse cotée et adresse non cotée.

Propriété : les intersections de ces 3 ensembles prises deux à deux sont vides.

Définition ④ :

... etc.

4-2- L'Alphabet :

Les programmes machines-outils sont écrits et codés par l'intermédiaire d'une machine à écrire. Nous distinguerons l'Alphabet de la machine-outil, l'Alphabet de la machine à écrire, puis l'Alphabet codé de la machine à écrire.

4-2-1- Alphabet de la machine-outil : noté $\mathcal{A}_{MO}$

1°) Définition :

C'est un ensemble fini de symboles formels (dits aussi caractères) "compréhensibles" par l'unité d'entrée de la machine-outil.

.../...

2°) Syntaxique :

$\langle \mathcal{A}_{MO} \rangle ::= \langle \text{adresse} \rangle | \langle \text{signe} \rangle | \langle \text{chiffre} \rangle |$
 $\langle \text{caractère auxiliaire} \rangle$

avec :

$\langle \text{adresse} \rangle ::= A | B | C | D | E | F | G | H | I | J | K | M | N |$
 $P | Q | R | S | T | U | V | W | X | Y | Z | : | \%$
 $\langle \text{signe} \rangle ::= + | -$
 $\langle \text{chiffre} \rangle ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9$
 $\langle \text{caractère auxiliaire} \rangle ::= BL | / | FB | (|)$

3°) L'application sémantique : ou assignation informationnelle fait correspondre à chaque adresse du langage machine-outil, une famille d'action du langage parlé de la gamme d'usinage et une seule..

4-2-2- L'alphabet de l'imprimante : noté $\mathcal{A}_{MI}$

1°) Définition :

C'est l'ensemble fini des caractères formels que peut imprimer la machine à "écrire", en particulier l'Alphabet Machine-outil.

2°) Propriétés syntaxiques :

Il comprend l'ensemble des caractères de l'Alphabet machine-outil et, d'autres caractères que nous n'aurons pas à préciser pour mener à bien l'étude de l'unité d'entrée de la machine-outil, puisqu'elle n'a pas à les connaître.

$$\mathcal{A}_{MO} \subset \mathcal{A}_{MI}$$

4-2-3- L'alphabet-codé :

1°) Définition :

Si l'on associe, à chaque caractère de l'alphabet de l'imprimante $\mathcal{A}_{MI}$, un état binaire et un seul, de n variables (dites aussi éléments) $e_1, e_2, \dots, e_n$ binaires, nous avons un alphabet codé, noté : $\mathcal{A}_{MI}^c$

2°) Codes :

L'on peut utiliser plusieurs codages, en particulier :

a) Le code de l'"Electronic Industries Association"
Réf. : EIA/RS 427 utilisé couramment.

b) Le code de l'"International Standards Organisation"
Réf. : ISO/TC 27 conseillé en particulier par les Normes françaises, commençant à être utilisé par les constructeurs.

.../...

4-3- Les Mots :

Nous avons deux sous-ensembles de mots :

- Les mots EN PROGRAMME, ce sont les mots qui sont compris et exécutés par la machine-outil.
- Les mots HORS PROGRAMME, ces mots ne sont pas compris ni exécutés par la machine-outil, elle n'aura pas à les lire. Généralement ce sont des commentaires en langage parlé.

$\langle \text{mot} \rangle ::= \langle \text{mot EN programme} \rangle \mid \langle \text{mot HORS programme} \rangle$

4-3-1- Les mots HORS programme :

1°) Définition :

Un mot hors programme est un ensemble ordonné, défini de symboles appartenant à l'Alphabet de la machine à écrire, caractères : (,et,) exclus, compris entre les caractères (,et,).

2°) Action machine :

Ils ne devront pas être exécutés par la machine-outil qui n'aura donc pas à les lire.

3°) Syntaxique :

- Ces mots sont à format variable quelconque
- $\langle \text{mot HORS programme} \rangle ::= \underline{a_i} \mid \underline{a_i, a_j \dots}$
avec $a_i, a_j \in \mathcal{A}_{MY}$, caractères : (,et,) exclus.

4-3-2- Les mots EN programme :

1°) Définition :

Un mot en programme est un ensemble ordonné, défini, de caractères appartenant à l'Alphabet machine-outil $\mathcal{A}_{MO}$, non compris entre les caractères (,et,).

2°) Action machine :

Les mots sont lus et exécutés par la machine-outil.

3°) Syntaxique :

- Ces mots sont de trois types :

$\langle \text{mot en programme} \rangle ::= \begin{array}{l} \langle \text{mot nonctuation} \rangle \mid \\ \langle \text{mot adresse cotée} \rangle \mid \\ \langle \text{mot adresse non cotée} \rangle \end{array}$

- Leur format est variable (format des entiers), mais pour chaque adresse, le format à une valeur maximale notée: N_i (i étant l'adresse considérée), voir planche N°4-1.

.../...

a) les mots ponctuation sont tels que :

$\langle \text{mot ponctuation} \rangle ::= \langle \text{Fin de bloc} \rangle | / | \langle \text{Blanc} \rangle$
 avec $\left[\begin{array}{l} \langle \text{Fin de bloc} \rangle ::= \text{FB} | \text{FB} \langle \text{Fin de Bloc} \rangle \\ \langle \text{Blanc} \rangle ::= \text{BL} | \text{BL} \langle \text{Blanc} \rangle \end{array} \right.$

b) les mots adresse cotée sont tels que :

$\langle \text{mot adresse cotée} \rangle ::= \langle \text{adresse cotée} \rangle \langle \text{Signe} \rangle \langle \text{entier} \rangle$
 avec $\left[\begin{array}{l} \langle \text{adresse cotée} \rangle ::= \text{A} | \text{B} | \text{C} | \text{D} | \text{E} | \text{I} | \text{J} | \text{K} | \text{P} \\ \quad \quad \quad \text{Q} | \text{R} | \text{U} | \text{V} | \text{W} | \text{X} | \text{Y} | \text{Z} \\ \langle \text{entier} \rangle ::= \langle \text{chiffre} \rangle | \langle \text{chiffre} \rangle \langle \text{entier} \rangle \end{array} \right.$

c) les mots adresse non cotée sont tels que :

$\langle \text{mot adresse non cotée} \rangle ::= \langle \text{adresse non cotée} \rangle \langle \text{entier} \rangle$
 avec $\left[\begin{array}{l} \langle \text{adresse non cotée} \rangle ::= \text{F} | \text{G} | \text{M} | \text{N} | \text{S} | \text{T} | \text{I} \\ \langle \text{entier} \rangle ::= \langle \text{chiffre} \rangle | \langle \text{chiffre} \rangle \langle \text{entier} \rangle \end{array} \right.$

d) Nota :

Les adresses sont telles que :

$\langle \text{adresse} \rangle ::= \langle \text{adresse cotée} \rangle | \langle \text{adresse non cotée} \rangle$

4°) L'application sémantique : fait correspondre à chaque mot du langage parlé machine-outil, une action élémentaire particulière du langage parlé de la gamme d'usinage (voir la planche n°4-1 précisant les applications sémantiques des mots).

L'on remarque que pour les mots fonction préparatoire et fonction auxiliaire, nous avons une action élémentaire par nombre entier associé à l'adresse G ou H, si ce nombre appartient aux tableaux.

Voir la planche n°4-2 précisant les Applications sémantiques des mots fonctions auxiliaires.

4-3-3- Mots en programme remarquables :

Les mots fonctions préparatoires et les mots fonctions auxiliaires se décomposant en familles fonctionnelles de mots suivant leur application sémantique ci-dessus.

.../...

1°) Les mots fonctions préparatoires :

$\langle \text{mot fonctions préparatoires} \rangle ::= \langle \text{sous-unité appelée} \rangle$
 $\langle \text{système de commande} \rangle | \langle \text{précision} \rangle | \langle \text{type de trajectoire} \rangle$
 $\langle \text{changement de vitesse d'avance} \rangle | \langle \text{espace choisi} \rangle$
 avec : $\left[\begin{array}{l} \langle \text{Type de trajectoire} \rangle ::= \langle \text{mise en position} \rangle | \\ \quad \langle \text{contournage} \rangle | \langle \text{Décalage d'origine} \rangle | \\ \quad \langle \text{cycles fixes} \rangle | \langle \text{correction d'outil} \rangle \\ \langle \text{Espace choisi} \rangle ::= \langle \text{axes choisis} \rangle | \langle \text{plan choisi} \rangle | \\ \quad \langle \text{tièdre choisi} \rangle \end{array} \right.$

2°) Les mots fonctions auxiliaires :

Voir planche 4 -2.

$\langle \text{mot fonctions auxiliaires} \rangle ::=$
 $\langle \text{fonctions auxiliaires programme} \rangle | \langle \text{préparation machine} \rangle |$
 $\langle \text{opération broche} \rangle$

4-4- Les Blocs :

4-4-1- Définitions :

Un bloc est un ensemble partiellement ordonné, fini, de mots en programme et éventuellement de mots hors programme, appartenant à l'ensemble des mots de la machine-outil.

Un bloc commence par le mot numéro de bloc et termine par le premier mot $\langle \text{fin de bloc} \rangle$ qui suit un mot du bloc, quel que soit ce mot.

4-4-2- Types de blocs :

Nous distinguerons deux types :

$\langle \text{Bloc} \rangle ::= \langle \text{bloc optionnel} \rangle | \langle \text{bloc non optionnel} \rangle$

1°) Les Blocs OPTIONNELS :

Ce sont des blocs que la machine lira et exécutera, ou ne lira et n'exécutera pas, en fonction du choix fait par le servant -ou opérateur) pilotant manuellement la machine-outil.

Ils sont tels qu'il existe dans ces blocs, le mot

$\langle \text{optionnel} \rangle ::= /$, en programme, suivant immédiatement le mot $\langle \text{numéro de bloc} \rangle$.

$\langle \text{mot optionnel} \rangle ::= \langle \text{Numéro de bloc} \rangle / \langle \text{mot ... mot} \rangle$
 $\quad \quad \quad \langle \text{fin de bloc} \rangle$

avec $\langle \text{mot ... mot} \rangle$ représentant des mots, en, et, ou, hors programme, mots $\langle \text{fin de bloc} \rangle$ $\langle \text{numéro de bloc} \rangle$ et $\langle \text{optionnel} \rangle$ exclus.

.../...

2°) Les blocs NON OPTIONNELS :

Ce sont des blocs que la machine lira et exécutera obligatoirement.

Ils sont tels qu'il n'existe pas dans ces blocs le mot
 <optionnel> :: = / , en programme, suivant immédiatement le
 mot numéro de bloc.

$$\langle \text{Bloc NOM optionnel} \rangle ::= \langle \text{Numéro de bloc} \rangle \langle \text{mot... mot} \rangle$$

$$\langle \text{Fin de bloc} \rangle$$

avec < mot... mot > représentant des mots en, et, ou hors programme, mots < numéro de bloc > et < fin de bloc > exclus.

3°) Remarque :

Les blocs optionnels et non optionnels ne diffèrent que de par la présence ou non, du mot <optionnel>::= / . Il nous suffira, afin de préciser les blocs, d'étudier, par exemple, les blocs non optionnels.

4-4-3- Syntaxique des blocs NON optionnels :

La formulation de la syntaxique des blocs NON optionnels étant très longue et lourde nous avons préféré donner le graphe maximum des syntaxiques de ces blocs (mots hors programmes exclus), boucles exclues.

Nous reprendrons plus en détail l'étude des blocs lors de l'établissement de l'ordinoگرامme des mises en mémoire des mots.

4-4-4- Blocs remarquables :

Nous distinguerons les blocs suivants qui sont remarquables de par leurs fonctions particulières dans le déroulement du programme machine-outil.

1°) Bloc de DEBUT de PIECE :

C'est un bloc qui comporte le mot :

< Mot numéro de bloc de début de pièce > ::= % 1

D'où la syntaxique :

<Aloc début pièce> ::= % 1 <mots...mots><fin de bloc>

2°) BLOC PROGRAMME TERMINE :

C'est un bloc qui comporte le mot :

< Mot fonction auxiliaire de FIN de BANDE > ::= M30

D'où la syntaxique :

$\langle \text{Bloc programme termin } \rangle ::= \langle \text{num ro bloc} \rangle \langle \text{mots... mots} \rangle \text{ M30}$
 $\langle \text{mots... mots} \rangle \langle \text{fin de bloc} \rangle$

.....

3°) Bloc PIECE TERMINEE :

C'est un bloc qui comporte le mot :

< mot fonction auxiliaire Pièce terminée > ::= M02

D'où la syntaxique :

[illegible]

4°) Bloc REPETITIF :

C'est un bloc qui comporte le mot :

$$\langle \text{numéro de bloc répétitif} \rangle ::= \text{!} \langle \text{entier} \rangle$$

D'où la syntaxique :

<Bloc répétitif> ::= ; <entier> <Mots... mots> <fin de bloc>

5°) Bloc d'usinage SEMI-AUTOMATIQUE :

C'est un bloc qui comporte le mot :

< mot fonction auxiliaire usinage semi-automatique > ::= M10

D'où la syntaxique :

< Bloc SEMI-AUTOMATIQUE > :: < Numéro de Bloc > < mots... mots > MJD
< mots... mots > < fin de bloc >

4-5- Les Programmes :

4-5-1-Notion de programme :

Un programme est un ensemble de données composé essentiellement de deux parties : les DIRECTIVES et les TACHES pièce. Il s'écrit :

```
<programme> ::= <Blanc> <Directives> <Blanc>
               <Tache pièce> <Blanc>
```

1°) Les DIRECTIVES permettent au préparateur, et à l'opérateur pilotant manuellement la machine-outil, de définir quels sont pour une pièce :

- a) le numéro et le nom d'Archive de la pièce
- b) les numéros des fiches de préparation d'usinage et des fiches d'usinage sur machine-outil.
- c) les numéros des fiches des montages d'usinage nécessaires
- d) les numéros des fiches de réglage des outils nécessaires et leur numéro d'indexation dans le casier porte-outil.

Ces données ne seront pas exécutées par la machine-outil elle n'aura pas à les comprendre. Elles formeront un ensemble qui sera un mot HORS programme particulier que nous n'étudierons pas plus.

```
<Directives> ::= <Fin de bloc> <mot HORS programme>  
                <fin de bloc>
```

...../.....

2°) Les TACHES pièces sont l'ensemble des informations en langage machine-outil et des commentaires associés en langage parlé, qui permettent respectivement, à la machine et à l'opérateur de réaliser la pièce.

C'est un ensemble fini de blocs appartenant à l'ensemble des blocs . Le dernier bloc de tâche pièce et seulement celui-là est : soit un « bloc programme terminé », soit un « bloc pièce terminée ».

Son format est variable (format des blocs et nombre des blocs).

Tâche pièce peut être de 4 types différents :

$\langle \text{Tâche pièce} \rangle ::= \langle \text{Tâche totale pièce} \rangle | \langle \text{Tâche début pièce} \rangle |$
 $\langle \text{Tâche milieu pièce} \rangle | \langle \text{Tâche fin de pièce} \rangle$

4-5-2- Types de programme :

Nous distinguerons quatre types de programme :

$\langle \text{programme} \rangle ::= \langle \text{Programme total pièce} \rangle |$
 $\langle \text{Programme début pièce} \rangle |$
 $\langle \text{Programme milieu pièce} \rangle |$
 $\langle \text{Programme fin pièce} \rangle$

classés en fonction du bloc initial et du bloc final de tâche pièce .

1°) Programme total pièce :

Un programme est dit programme total pièce si tâche pièce est telle que son premier bloc soit un bloc début pièce, et son dernier soit un bloc pièce terminée.

$\langle \text{programme total pièce} \rangle ::= \langle \text{Blanc} \rangle \langle \text{Directives} \rangle \langle \text{Blanc} \rangle$
 $\langle \text{tâche totale pièce} \rangle \langle \text{Blanc} \rangle$
 avec $\langle \text{tâche totale pièce} \rangle ::= \langle \text{fin de bloc} \rangle \langle \text{bloc début pièce} \rangle$
 $\langle \text{bloc} \dots \text{bloc} \rangle$
 $\langle \text{Bloc pièce terminée} \rangle$

2°) Programme début pièce :

Un programme est dit programme début pièce si tâche pièce est telle que son premier bloc soit un bloc début pièce et son dernier soit un bloc programme terminé.

$\langle \text{programme début pièce} \rangle ::= \langle \text{Blanc} \rangle \langle \text{Directives} \rangle \langle \text{Blanc} \rangle$
 $\langle \text{Tâche début pièce} \rangle \langle \text{Blanc} \rangle$
 avec $\langle \text{tâche début pièce} \rangle ::= \langle \text{Fin de bloc} \rangle \langle \text{Bloc début pièce} \rangle$
 $\langle \text{Bloc} \dots \text{bloc} \rangle$
 $\langle \text{Bloc programme terminé} \rangle$

.../...

3°) Programme milieu de pièce :

Un programme est dit programme milieu de pièce, si tâche pièce est telle que son premier bloc ne soit pas un bloc début pièce et son dernier soit un bloc programme terminé.

$\langle \text{Programme milieu pièce} \rangle ::= \langle \text{Blanc} \rangle \langle \text{Directives} \rangle \langle \text{Blanc} \rangle$
 $\langle \text{tâche milieu pièce} \rangle \langle \text{Blanc} \rangle$
 avec $\langle \text{tâche milieu pièce} \rangle ::= \langle \text{Fin de bloc} \rangle$
 $\langle \text{bloc dif. début pièce} \rangle \langle \text{Blocs... Blocs} \rangle$
 $\langle \text{bloc programme terminé} \rangle$

4°) Programme fin de pièce :

Un programme est dit programme fin de pièce, si tâche pièce est telle que son premier bloc ne soit pas un bloc début de pièce et son dernier soit un bloc pièce terminée.

$\langle \text{Programme fin de pièce} \rangle ::= \langle \text{Blanc} \rangle \langle \text{Directives} \rangle \langle \text{Blanc} \rangle$
 $\langle \text{Tâche fin de pièce} \rangle \langle \text{Blanc} \rangle$
 avec $\langle \text{tâche fin de pièce} \rangle ::= \langle \text{Fin de bloc} \rangle$
 $\langle \text{bloc dif. début pièce} \rangle \quad \langle \text{Blocs ... blocs} \rangle$
 $\langle \text{bloc pièce terminée} \rangle.$

4-5-3- Sous-programmes remarquables :

Nous appellerons sous-programme une partie d'un programme formée par un ensemble connexe de mots appartenant à la tâche pièce.

Nous distinguerons trois familles de sous-programmes :

1°) Famille des sous-programmes dits Stop optionnel ou Stop NON optionnel :

a) Sous-programmes Stop optionnel :

C'est un sous-programme comprenant tous les mots appartenant à tâche pièce, NON précédés du mot Stop non optionnel, c'est à dire : soit précédés de Stop optionnel, soit précédé de ni l'un ou ni l'autre.

$\langle \text{Sous-programme stop optionnel} \rangle ::= \langle \text{mot stop optionnel} \rangle$
 $\langle \text{mots... mots} \rangle \langle \text{mot stop non optionnel} \rangle \langle \text{mot début tâche pièce} \rangle$
 $\langle \text{mots... mots} \rangle \langle \text{mot stop non optionnel} \rangle$

b) Sous-programme Stop NON optionnel :

C'est un sous-programme comportant tous les mots appartenant à tâche pièce, précédés du mot Stop NON optionnel exclus et non précédés du mot Stop optionnel inclus.

$\langle \text{Sous-programme stop NON optionnel} \rangle ::=$
 $\langle \text{mot stop NON optionnel} \rangle \langle \text{mots... mots} \rangle \langle \text{mot stop optionnel} \rangle$

.../...

2°) Famille des sous-programmes dit Répétitifs ou non répétitifs :

a) Sous-programmes REPETITIF :

C'est un sous programme formé par l'ensemble connexe des blocs, dont le premier bloc est le $(2n + 1)^{\text{ème}}$ bloc répétitif du programme et dont le dernier bloc est le $(2n + 2)^{\text{ème}}$ bloc répétitif du programme.

< Sous-programme répétitif > ::= < $2n + 1$ bloc répétitif >

< bloc ... bloc > < $2n + 2$ bloc répétitif >

b) Sous-programme NON REPETITIF :

C'est un sous-programme formé par l'ensemble connexe des blocs :

Dont le premier bloc est : soit le premier bloc du programme (s'il n'est pas répétitif), soit le bloc suivant le $(2n + 2)^{\text{ème}}$ bloc répétitif.

Dont le dernier bloc est le bloc qui précède le $(2n + 3)^{\text{ème}}$ bloc répétitif du programme.

< Sous-programme répétitif > ::= < $2n + 2$ bloc répétitif > < Bloc ...

... Bloc > < $2n + 3$ bloc répétitif > |

< Bloc début programme > < Blocs...blocs > < 1er bloc répétitif >

3°) Famille des sous-programmes dits optionnels ou obligatoirement (non optionnels) :

a) sous-programme OPTIONNEL :

C'est un sous-programme formé par l'ensemble connexe des blocs OPTIONNELS compris entre deux blocs NON OPTIONNELS.

< Sous-programme OPTIONNEL > ::=

< Bloc NON optionnel > < Bloc optionnel ... Bloc optionnel >

< Bloc non optionnel >

b) Sous-programme NON OPTIONNEL :

C'est un sous-programme formé par l'ensemble connexe des blocs NON OPTIONNELS compris entre deux blocs OPTIONNELS.

< Sous-programme NON OPTIONNEL > ::= < Bloc optionnel >

< Bloc NON optionnel ... Bloc NON optionnel >

< Bloc optionnel >

* Les parties soulignées n'appartiennent pas à la proposition syntaxique

.../...

4-5-4- Ruptures de séquences programme dites ruptures :

Nous appelons rupture de séquence de programme tout caractère qui suit, dans le sens du déroulement du programme l'un des mots, blocs, sous-programme précisé ci-après.

< Rupture de programme > ::= < Rupture d'avance programme > |
< Rupture de Retour programme >

Nous avons indiqué l'emplacement de ce caractère par le symbole 

1°) Dans le sens Avance du programme (lecture normale de gauche à droite) nous avons :

< Rupture Avance > ::= < Rup. Av. Début programme >
< Rup. Av. Bloc répétitif > < Rup. Bloc optionnel >
< Rup. usinage semi-automatique >

avec : $\left[\begin{array}{l} \text{< Rup. Av. Début programme > ::= < Blanc > < Directives >} \\ \hspace{15em} \text{< Blanc > < Tâche pièce >} \\ \text{< Rup. Av. Bloc répétitif > ::= < Sous-programme répétitif >} \\ \text{< Rup. Bloc optionnel > ::= < Numéro bloc > / < mots... mots >} \\ \text{< Rup. usinage semi-auto. > ::= < Bloc usinage semi-automatique >} \end{array} \right.$

2°) Dans le sens Retour du programme (lecture inverse, de droite à gauche) nous avons :

< Ruptures Retour > ::= < Rup. Re. Début programme > |
< Rup. Re. Bloc répétitif >

avec : $\left[\begin{array}{l} \text{< Rup. Re. Début programme > ::= < Blanc > < Directive >} \\ \hspace{15em} \text{< Blanc > < Tâche pièce >} \\ \text{< Rup. Re. Bloc répétitif > ::= < Sous-programme répétitif >} \end{array} \right.$

4-6- Les bandes programmes codées :

4-6-1- Rappel de l'Alphabet codé machine à écrire :

Nous avons une application bijective entre l'Alphabet de la machine à écrire : $\mathcal{A}_{MI}$ et l'Alphabet codé de la machine à écrire $\mathcal{C}\mathcal{A}_{MI}$.

A chaque caractère de $\mathcal{A}_{MI} \longleftrightarrow$ un état binaire de n variables (e_1 à e_n c'est à dire 8 dans le cas de codes ISO et EIA étudiés, choisi parmi : $\{0, 1\}^n$).

Nous noterons que le code ISO est pair, alors que le code EIA est impair. Pair signifiant que chaque caractère codé comporte un nombre pair de 1, donc de zéro si n est pair (en particulier pour 8).

.../...

4-6-2- Programme codé :

Un programme dont chaque caractère est écrit par l'état des n variables binaires déterminées par $\alpha_{MI} \iff \beta_{MI}$ sera dit un programme codé.

Ex.: en code ISO M30 s'écrit :

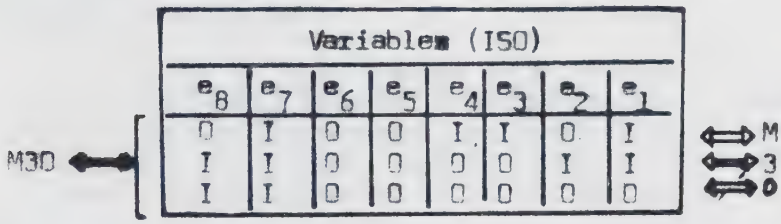
M30 $\iff$ 0 0 I 0 I I 0 I $\uparrow$ I I 0 0 0 0 I I $\uparrow$ I I 0 0 0 0 0 0

avec : $\uparrow$ indication de séparation des caractères codés.

4-6-3- Bande programme codée :

Si chaque caractère codé d'un programme est écrit en colonne, variables binaires e_i en colonnes $\forall i \in n$, dans un ordre déterminé fixe, nous dirons que le programme est écrit en bande programme codée.

Ex.: en code ISO , M30 s'écrit :



4-7- Conclusions :

4-7-1- Remarques relatives aux langages I.S.O. et E.I.A. :

1°) L'étude de la syntaxique des blocs nous a conduit à dégager des variables métalinguistiques particulières pour les mots fonctions préparatoires et les mots fonctions auxiliaires. Nous ne discuterons pas du choix des entiers de ces différents mots car il a été imposé aux langages par les machines existantes sur le marché.

2°) Nous remarquerons que le langage a évolué comme les caractéristiques des machines (nombre d'axes, de têtes d'usinage, de systèmes d'arrosages... etc), c'est à dire que l'on a rajouté un mot ou même une adresse lorsque l'extension de la machine le nécessitait.

Afin de rendre systématique la syntaxique d'un bloc nous avons introduit une variable métalinguistique supplémentaire dans les mots fonctions préparatoire permettant d'appeler des sous-unité d'usinage (par exemple la table, la tête d'usinage n°2...)

Nous n'avons pas osé introduire les simplifications qui s'imposaient dans les mots et même les adresses, après l'introduction de ces nouvelles fonctions préparatoires.

.../...

3°) Nous avons introduit, dans les mots fonctions auxiliaires de programmation deux mots particuliers, à savoir :

a) L'autorisation de passer en usinage semi-automatique :

permettant : - de émettre, en cours d'usinage, la machine aux cotes initiales zéro (calage d'origine)
- d'effectuer des usinages manuels au "milieu" d'usinage automatique

b) Le mot Stop NON optionnel : Il permet de ne pas arrêter l'usinage et, ou, la lecture du programme à n'importe quel instant (par exemple : l'arrêt de l'usinage au milieu d'une passe de finition est très déconseillé : marquage de la pièce). Il évite (nous le verrons par la suite) d'avoir à programmer pour chaque bloc dit Stop optionnel, le mot Stop optionnel.

4°-7-2- Conclusions :

L'étude de la syntaxique d'un langage MACHINE-OUTIL nous a permis de définir le langage de la machine-outil, d'en préciser les propriétés et les contraintes, de formuler aussi de manière précise, l'ensemble des données possibles que comporte un programme, en les classant par familles syntaxiques identiques.

Nous en déduirons, les logigrammes généraux de l'unité d'entrée et l'ensemble des sous-unités logiques permettant de "traiter" les programmes.

5 - ORDINOGRAMMES DE L'UNITE D'ENTREE

	pages :
5-1- <u>Introduction aux ordinogrammes</u>	80
5-1-1- Symbolisations utilisées	80
5-1-2- Principes d'établissement d'un ordinogramme	84
5-2- <u>Ordinogrammes globaux de la machine</u>	89
5-2-1- Ordinogramme de démarrage (P1)	89
5-2-2- Ordinogramme d'usinage automatique (P3)	90
5-3- <u>Développement de la phase de lecture d'un bloc en usinage automatique</u>	92
5-3-1- Ordinogramme de la lecture d'un bloc	92
5-3-2- Ordinogramme de la lecture de un caractère (SP2)	93
5-4- <u>Ordinogrammes de l'unité d'entrée de la machine à réaliser.</u>	104
5-4-1- Syntaxique du langage de M01	104
5-4-2- Méthode d'élimination des phases d'évolutions	105
5-4-3- Etablissement des ordinogrammes de la machine M01.	106

Introduction :

Nous nous proposons d'établir, dans ce chapitre, les ordinogrammes de l'unité d'entrée de la machine à réaliser (voir les planches 5-1 à 5-9 jointes).

La méthode que nous suivons est essentiellement heuristique. Après avoir introduit les symbolisations graphiques des schémas blocs nous indiquerons les principes essentiels d'établissement des ordinogrammes.

En fonction des états globaux de la machine, des ordres manuels particuliers (chapitre 3) et de l'étude syntaxique des programmes codés (chapitre 4) nous établirons les ordinogrammes des phases d'évolutions globales de la commande numérique (ordinogrammes P1 et P3).

Nous reprendrons chaque phase d'évolution, nous la développerons cela nous conduira, de proche en proche à l'ordinogramme détaillé de l'unité d'entrée d'une commande numérique comportant toutes les propositions du langage codé I.S.O.

Puis après avoir précisé l'Alphabet et les mots du langage de la machine à réaliser, nous simplifierons les ordinogrammes précédents. Nous aboutirons aux ordinogrammes (indiqués M01) de la machine que nous devons construire.

Nous ne développerons pas dans ce texte toutes les phases d'évolutions, nous nous limiterons à la phase de lecture d'un bloc (ordinogramme P3) que nous détaillerons jusqu'à la lecture de un caractère (ordinogramme SP2).

5-1- Introduction aux ordinogrammes : symbolisation et règles :

5-1-1- Symbolisations utilisées :

Nous avons utilisé les symbolisations employées couramment en programmation, en nous efforçant de suivre les conseils de la Norme Française AFNOR n° Z-67-010. Nous distinguerons :

1°) Les blocs diagrammes des états ou évolutions d'états de la machine :

Aucune action de l'opérateur ne doit être représentée par ces blocs, même si elle implique un état, ou une évolution d'états de la machine.

Désignations et Fonctions essentielles	Symboles
- <u>Opérations ou groupes d'opérations</u> caractérisant un segment (sauf tous les blocs diagrammes ci-après). <u>Utilisé</u> pour les états de la commande	
- <u>Sous-programme</u> <u>Utilisé</u> pour les évolutions d'états de la commande, un ensemble d'évolutions d'états correspondant à une action physique déterminée.	
- <u>Bloc de décision</u>, dit alternative <u>Utilisé</u> pour les embranchements, les ruptures de séquences déterministes fixées par un état interne passé de la machine. <u>Ne pas utiliser</u> pour les choix de l'opérateur.	
- <u>Commentaire</u> <u>Utilisé</u> pour compléter, les informations contenues dans un bloc diagramme. Le commentaire ne fait pas partie de l'ordinogramme, il est destiné au lecteur et doit permettre à celui-ci de mieux saisir le sens de l'ordinogramme.	

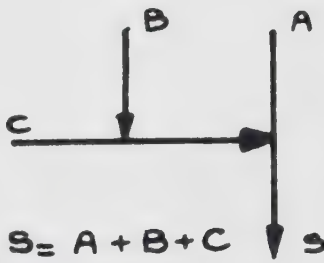
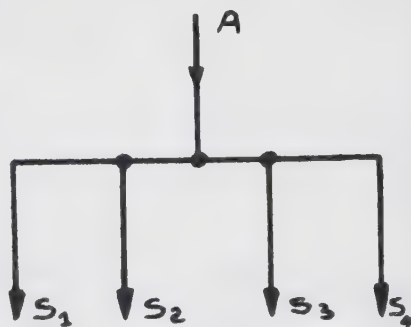
2°) Les blocs diagrammes des actions de l'opérateur :

Toute action de l'opérateur impliquant un état ou une évolution d'états de la machine sera symbolisée par des blocs diagrammes différents de ceux des états ou des évolutions d'états de la machine impliqués par d'autres états.

Désignations et Fonctions essentielles	Symboles
- <u>Bloc d'action sur une entrée manuelle.</u> Utilisé pour l'action sur une ou plusieurs entrées manuelles (simultanées). L'évolution de la commande est alors représentée par les blocs diagrammes précisés précédemment.	
- <u>Bloc d'action de sous-programme d'entrée manuelle.</u> Utilisé pour représenter une ou plusieurs actions manuelles de l'opérateur ainsi que les états ou évolutions d'états impliquées dans la machine. Un tel sous-programme comporte au moins une action manuelle en tête.	
- <u>Bloc de choix.</u> Utilisé pour représenter un point de branchement ou une rupture de séquence telle que la séquence suivante soit laissée au libre choix de l'opérateur. Le choix des séquences suivantes possibles sera déterminé par les blocs diagrammes d'action qui suivent ce bloc de choix. L'opérateur pourra effectuer le choix en fonction de critères qui pourront être précisés dans le bloc de choix.	

3°) Les lignes de liaisons :

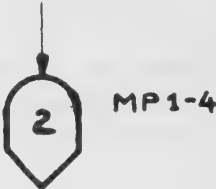
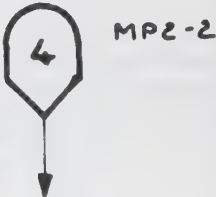
Ce sont des segments orientés qui relient les blocs d'une même page entre eux. L'on utilise les liaisons ci-dessous.

Désignations et Fonctions essentielles	Symboles
- <u>Segments</u> (orientés) <u>Utilisé</u> pour relier deux blocs diagrammes. Si l'action ou l'évolution précisée dans le bloc amont A est terminée, l'évolution ou l'action précisée dans le bloc aval B commencera.	
- <u>Connections OU de n segments</u> <u>Utilisées</u> pour relier n blocs à 1 bloc déterminé. Si l'une des actions ou évolutions précisée dans un bloc amont A est terminée, l'action ou évolution aval commencera.	 $S = A + B + C$
- <u>Connections ET de n segments</u> <u>Utilisées</u> pour relier 1 bloc à n blocs déterminés. Si l'action ou l'évolution précisée dans le bloc amont A est terminée, les actions ou évolutions précisées dans les n blocs aval, commenceront.	

.../...

4°) Les blocs diagrammes de renvoi:

Afin de faciliter la lecture ou la représentation des ordinogrammes, il est d'usage d'introduire des blocs diagrammes de renvoi des lignes de liaison, soit à l'intérieur d'une page soit d'une page à une autre page.

Désignations et Fonctions essentielles	Symboles
- <u>Point de renvoi dans la page.</u> <u>Indexé</u> par le numéro du renvoi dans la page. Ce numéro est identique à celui du point d'origine correspondant.	
- <u>Point d'origine dans la page.</u> <u>Indexé</u> par le numéro d'origine dans la page. Ce numéro est identique à celui du point de renvoi correspondant.	
- <u>Point de renvoi Hors de la page.</u> <u>Indexé</u> par : - le numéro d'ordre du renvoi de la page; ex. 2 - les numéros des pages des points où va le segment ainsi que le numéro d'ordre du point origine hors de la page correspondant. Ex. le point de renvoi va en page MP1 au point d'origine n°4	
- <u>Point d'origine hors de la page.</u> <u>Indexé</u> par : - le numéro d'ordre du point origine de la page ; ex. 4 - les numéros des pages des points d'où vient le segment ainsi que le numéro d'ordre du point de renvoi hors de la page correspondant Ex. le point d'origine vient de la page MP2 du point 2	

5°) Les blocs diagrammes et repères auxiliaires :

Lors de l'étude de plus en plus détaillée des ordinogrammes, l'on développe au fur et à mesure des sous-programmes d'un programme déjà étudié.

Afin de faciliter la lecture de ces développements,

- chaque zone développée dans un sous-programme est entourée d'un trait mixte et, est repérée par le numéro du sous-programme développé.
- chaque segment traversant ces traits mixtes est repéré par une circonférence indexée par un numéro d'ordre.

5-1-2- Principes d'établissement d'un ordinogramme :

Etablir l'ordinogramme d'une machine logique revient :

- A déterminer des $\langle \text{phases d'évolution} \rangle ::= \langle \text{état} \rangle | \langle \text{évolutions d'états} \rangle | \langle \text{action manuelle et évolution d'états} \rangle$, correspondant à des applications déterminées dans la sémantique de la machine (l'ensemble des actions qu'elle devra être capable d'effectuer).

- A déterminer les enchainements de ces phases d'évolutions entre elles en fonction :

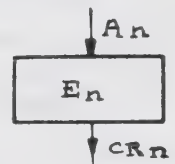
- du temps
- des actions manuelles de l'opérateur
- d' $\langle \text{états} \rangle$ atteints ou non pendant les phases d'évolutions antérieures.

- Puis à décomposer des phases d'évolution données en phases d'évolutions plus détaillées, en introduisant les nouveaux enchainements entre ces phases.

1°) Notion de phase d'évolutions :

a) Caractéristiques d'une phase d'évolutions :

Soit une phase d'évolutions E_n ci-contre, à une époque donnée ; l'on peut écrire que la phase d'évolutions considérée :



① Commence, si l'Autorisation d'évolution A_n est vraie.

② Est en cours, si l'on est en séquences d'évolutions SqE_n , c'est à dire si en fonction du temps l'on décrit une suite d'évolutions prévue dans l'ensemble des évolutions possibles dans cette phase d'évolutions.

.../...

- ③ Se termine, si la séquence d'évolutions prévue vient de s'effectuer ; nous dirons que son compte-rendu d'évolutions CR_n est vrai.
- ④ N'est ni en commencement, ni en cours, ni ne se termine, c'est à dire qu'elle a impliqué une autre séquence d'évolutions dans une autre phase d'évolutions. Nous dirons que nous sommes en séquence d'évolution bloquée dans la phase considérée.

b) Phases d'évolutions simultanées :

Nous dirons que des phases d'évolutions sont simultanées si elles sont en cours, en même temps, sur un intervalle de temps donné.

c) Indexation d'une phase d'évolutions :

Une phase d'évolutions est dénommée par l'ensemble des actions qu'elle implique dans la sémantique de la machine.

Afin de distinguer des phases d'évolutions ayant la même application sémantique, mais se produisant à des instants différents, et impliquées par des phases d'évolutions antérieures différentes, nous indexerons chaque bloc diagramme des $\langle$ phases d'évolutions $\rangle$ par une triple indexation :

$\langle$ Entier $\rangle$ $\langle$ Lettre minuscule $\rangle$ $\langle$ entier $\rangle$

$\langle$ entier $\rangle$: numéro de la page de l'ordinogramme
 $\langle$ lettre minuscule $\rangle$: précise la ligne dans laquelle se trouve le bloc diagramme.

$\langle$ entier $\rangle$: précise la numéro d'ordre sur la ligne considérée (en commençant par la gauche).

2°) Notion d'enchaînement des phases d'évolutions:

Considérons une époque t , où la phase d'évolution (Ex: Ea_2) vient de se terminer et supposons que l'on connaisse toutes les phases d'évolutions et leur ordre de succession dans le temps, décrites avant l'époque t .

a) Admettons que pendant la phase d'évolution Ea_2 considérée, il soit apparu des $\langle$ états $\rangle$ particuliers $\{Qa_2\}$ impliqués par des $\langle$ actions manuelles $\rangle$, des $\langle$ entrées programme $\rangle$ et la séquence des phases d'évolutions précédentes. Notre phase d'évolutions se termine, mais dans des états possibles différents.

.../...

Cette phase une fois terminée peut impliquer une phase d'évolutions au temps t^+ , parmi un ensemble de phases possibles { Ex. : Eb_1, Eb_2, Eb_3 } suivant que l'on a atteint ou pas les <états>² appartenant à $\{Qa_2\}$;
Ex.: $\{Qa_2\} = \{A, B\}$

Exemple :

A une époque donnée, si CRa_2 est vrai, et que pendant la phase d'évolution Ea_2 , l'on a atteint l'état A et l'état B, l'autorisation d'évolution Ab_1 sera VRAIE, la phase d'évolution Eb_1 commencera.

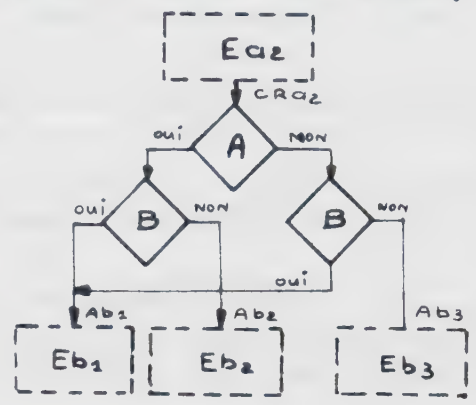


figure 1.

b) Les phases d'évolutions impliquées par Ea_2 peuvent être aussi fonction :

- Des <actions manuelles> que l'opérateur doit effectuer lorsque la phase Ea_2 est terminée.
- Des < états > apparus lors des phases d'évolution Ea_2 , que nous appellerons des prédécesseurs.

Exemple :

Reprenons l'exemple précédent, [figure 1] et introduisons un bloc de décision [$\langle \text{état} \rangle$: C, apparu pendant une phase Ef_4 précédente] et deux < actions manuelles > possibles pouvant impliquer des phases d'évolution lorsque Ea_2 est terminée.

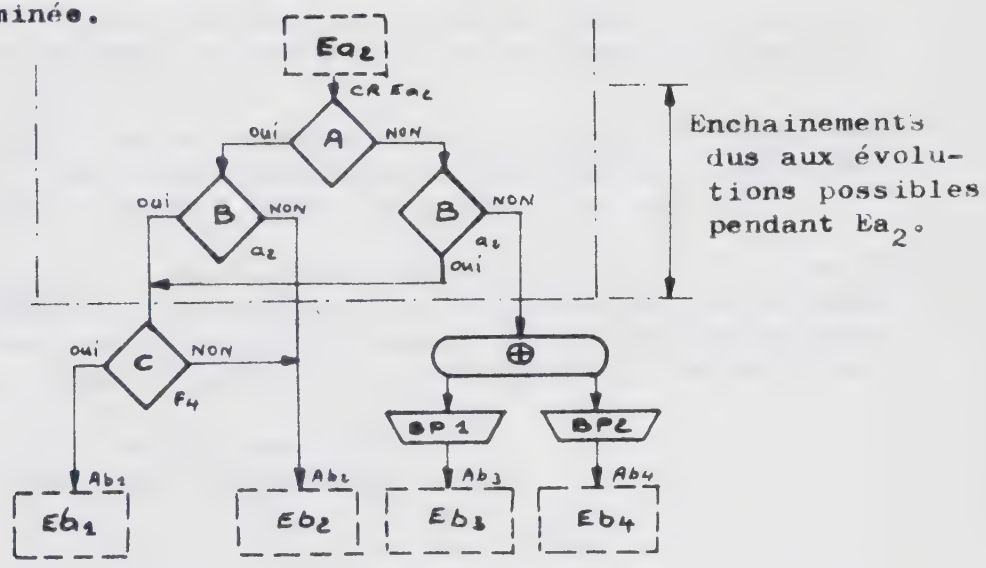


figure 2.

La phase Eb_3 commencera si à l'époque t , CRa_2 est VRAI, que l'on n'a pas atteint A, et pas atteint B pendant la phase d'évolution Ea_2 , et que l'opérateur effectue l'action manuelle BP_2/...

c) Assignation informationnelle:

Rappelons que nous avons choisi l'assignation informationnelle naturelle suivante, pour une proposition A à laquelle nous associerons une valeur booléenne A ou $\overline{A}$.

Si A est VRAI, ou atteint (OUI) ou effectué, on associe la valeur booléenne notée A .

Si A est FAUX, ou NON atteint, ou NON effectué on associe la valeur booléenne notée $\overline{A}$.

d) Equations booléennes d'un enchainement :

C'est l'ensemble des équations booléennes des autorisations d'évolutions impliquées par une phase d'évolutions terminée.

Exemple : Reprenons la figure ② , il vient :

$$\left[\begin{array}{l} Ab_1 = CRA_2 \circ B \circ C \\ Ab_2 = CRA_2 \circ (B \circ \overline{C} + A \circ \overline{B}) \\ Ab_3 = CRA_2 \circ \overline{A} \circ \overline{B} \circ BP_1 \\ Ab_4 = CRA_2 \circ \overline{A} \circ \overline{B} \circ BP_2 \end{array} \right.$$

e) Evolutions impliquées simultanées :

Nous dirons que deux phases évolutions Eb_1 et Eb_2 impliquées par une même évolution CRA_2 sont simultanées s'il existe une époque t , telle que Ab_1 et Ab_2 soient VRAIES.

C'est à dire que l'intersection des équations booléennes des autorisations d'évolutions ne soit pas vide.

Nota : Seule la sémantique des phases d'évolutions Eb_1 , Eb_2 peut nous permettre de savoir si ces deux phases peuvent être simultanées ou non.

Exemple : Reprenons l'ordinogramme de la figure 2 et recherchons les évolutions impliquées simultanées possibles. Pour ce faire, établissons le tableau de Karnaugh des fonctions Ab_1 , Ab_2 , Ab_3 , en indiquant par minterm :

$$\left[\begin{array}{ll} 1 & \text{si } Ab_1 \text{ est VRAI} \\ \text{Rien} & \text{si } Ab_1 \text{ est FAUX} \end{array} \right. \quad \dots / \dots$$

		2	2	1	1	2	
3		2	2	1	1	2	3
2,3		2	2	1	1	2	2,3
2		2	2	1	1	2	2

Les évolutions Eb_2 et Eb_3 sont simultanées pour le minterm $CRA_1 \cdot \overline{A} \cdot \overline{B} \cdot BP1 \cdot BP2$.

Dans le cas où elles ne peuvent l'être, on peut soit interdire ces minterms en écrivant les actions manuelles $BP1 \cdot \overline{BP2}$ et $BP2 \cdot \overline{BP1}$ (noté $\oplus$ dans le bloc de choix), soit choisir une action prioritaire.

f) Non évolution :

Nous n'aurons pas d'évolutions impliquées pour tout minterm n'impliquant pas d'évolution.

Nous distinguerons deux cas :

- Stabilité sur un bloc de choix : Si l'on n'effectue pas les actions manuelles situées sur un trajet issu d'un bloc de choix.

Ex.: les minterms $\overline{A} \cdot \overline{B} \cdot \overline{C} \cdot \overline{BP1} \cdot \overline{BP2}$ et $\overline{A} \cdot \overline{B} \cdot C \cdot \overline{BP1} \cdot \overline{BP2}$ sont des minterms de stabilité sur bloc de choix.

Pour toute action manuelle prévue il existe au moins un minterm de stabilité sur bloc de choix.

- Panne logique : Pour tout ensemble de minterms indépendant des actions manuelles, n'impliquant pas d'évolutions nous dirons que nous avons panne logique.

Ex.: pour $\overline{A} \cdot \overline{B} \cdot \overline{C}$, nous n'avons pas d'évolutions ; cela correspond à des états interdits, à l'époque t , des blocs des décisions A, B, C.

5 -2- Ordinogrammes globaux de la machine :

En fonction des états globaux de la programmation manuelle, des choix effectués par l'opérateur et des informations du programme codé, nous allons décrire les évolutions de la machine. Nous distinguerons deux phases essentielles:

1°) Le démarrage de la machine et la mise aux conditions initiales de la logique de la commande, de la bande programme codée, des axes de la machine-outil.

2°) L'usinage d'une pièce effectué en fonction des données de la bande programme codée.

5 -2-1- L'ordinogramme de démarrage et de mise aux conditions initiales : (Voir ordinogramme P1, page 89 bis)

1°) Mise sous tension puis mise aux conditions initiales de la commande de la machine.

L'on effectue la mise sous tension de la logique électronique, de la logique électrotechnique, la mise aux états initiaux des fonctions mémoire de toute la logique en régime forcé suivie de la mise en condition de tous les circuits de puissance (électrotechnique, hydraulique, régulations).

Le système étant aux conditions initiales l'opérateur peut choisir, à son gré, la mise aux conditions initiales de la bande programme codée ou des axes de la machine-outil.

2°) Mise aux conditions initiales de la bande programme codée :

La bande programme codée doit, en mise en place bande s'arrêter en début de tous les programmes se trouvant sur cette bande c'est à dire, sur :

```
<Rupture début programme > ::=
    <Rup. Avance début programme > |
    <Rup. Retour début programme >
```

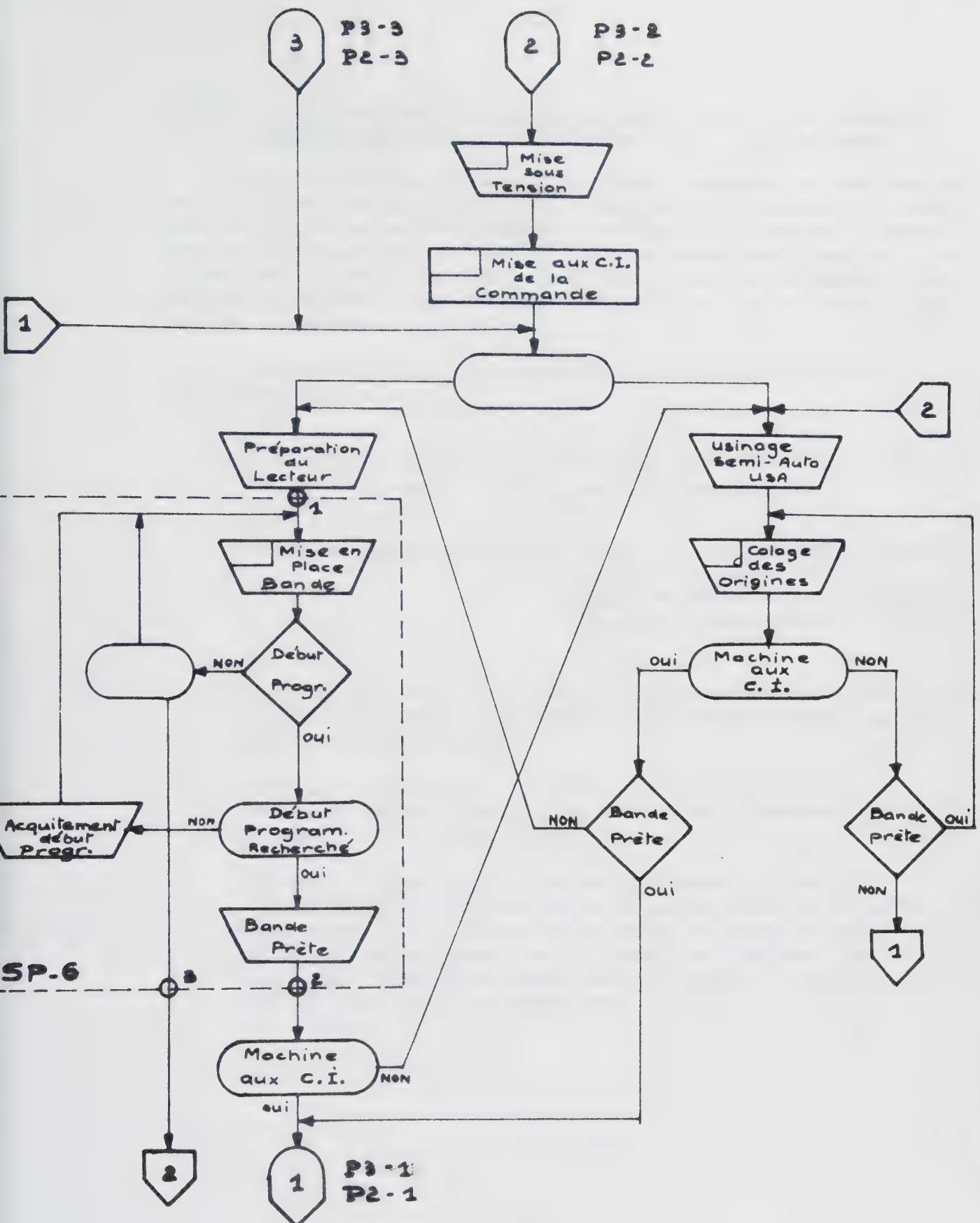
Dans cet état, l'opérateur choisit si le lecteur est au début du programme recherché ou non. Si oui, il indique au système que la bande (le programme) est prête. Si NON il continue la mise en place bande.

3°) Mise aux conditions initiales des axes de la machine-outil :

L'opérateur met en coïncidence les origines de la pièce et celles des axes de la machine, en déplaçant les axes par affichage de cotes à l'aide de la programmation manuelle.

4°) Nota : Nous ne détaillerons pas plus cet ordinogramme qui est considéré comme l'énoncé des possibilités que doit avoir la machine.

.../...



Ordinogramme global

Démarrage - Préparation du lecteur
Mise aux origines des axes de la machine

Ordinogramme
P1

5-2-2- L'ordinogramme d'usinage d'une pièce en fonction des données de la bande programme (voir ordinogramme P2W page 90 bis)

Après avoir mis la bande en place, indiqué au système que la bande est prête et calé les origines de la machine, l'opérateur programme manuellement le système en usinage automatique. Le système lit et "usine" le programme codé, sauf si l'on a une rupture de programme conduisant à une programmation manuelle différente de l'usinage automatique, ou si l'on a une fin de programme.

L'ensemble des ruptures de l'action < lit et usine un programme codé > est :

```
[ < Rupture programme usinage semi-automatique > ::=
    < Bloc usinage semi-automatique >

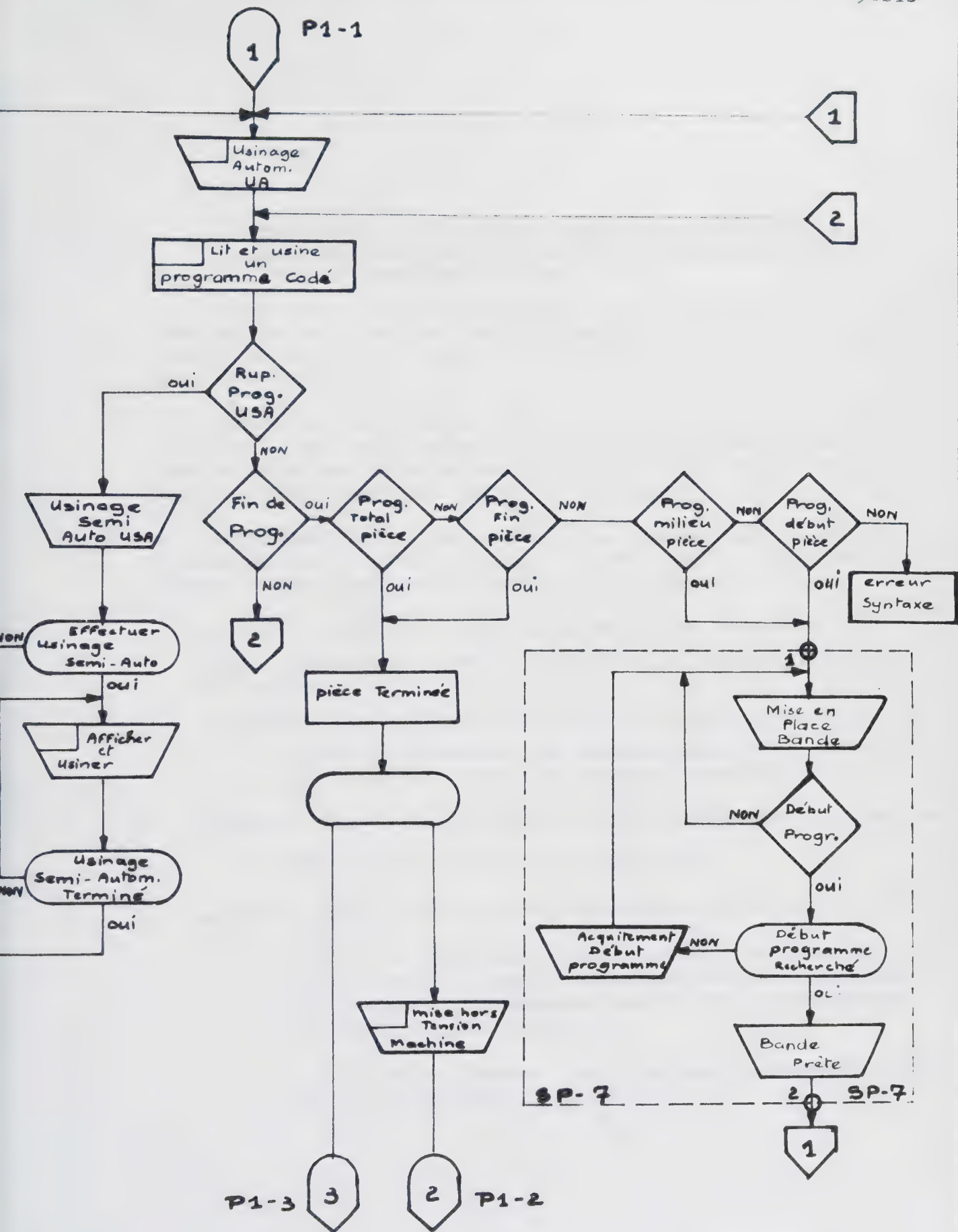
    < Fin de programme > ::= < Rupture tâche totale pièce > |
                                < Rupture tâche fin pièce > |
                                < Rupture tâche milieu pièce > |
                                < Rupture tâche début pièce > ]
```

1°) Dans le cas de la < Rup. programme usinage semi-auto >, l'opérateur devra programmer manuellement l'état global usinage semi-automatique.

2°) Dans le cas des < fin de programme > type :
< Rupture tâche totale pièce > ou de < Rupture tâche fin de pièce >, la pièce est terminée.

3°) Dans le cas des < fin de programme > type :
< Rupture tâche milieu de pièce > ou de < Rupture tâche début pièce >, l'opérateur devra changer la bande programme et rechercher les informations du début de la tâche pièce suivante. Cela fait, il reprendra le cycle de lecture et d'usinage du programme.

.../...



Ordinogramme global (Provisoire)
usinage automatique avec Ruptures de
programme

Ordinogramme
P2 W

Reprenons l'ordinogramme précédent en développant le sous-programme < lit et usine un programme codé > .
(Voir Ordinogramme P3 ,page 91bis)

L'on peut, dans la conception dite : série, lire un bloc puis usiner ce bloc, ou dans la conception dite : parallèle, lire le bloc n^o:n et usiner simultanément le bloc n^o:n-1, lu précédemment.

Bien que moins rapide, nous retiendrons la conception dite série, car elle conduit à une solution plus économique (pas de mémoires intermédiaires), après avoir admis que la machine-outil s'arrête pendant la lecture des blocs.

L'ensemble des ruptures de programme suivant l'opération d'usinage bloc terminée, sera l'ensemble des ruptures de programme se trouvant en fin de bloc, c'est à dire :

< Rupture bloc pièce terminée > $\Rightarrow$ pièce terminée

(l'opérateur peut arrêter la machine ou commencer une autre pièce).

< Rupture bloc programme terminé > $\Rightarrow$ programme terminé,
pièce non terminée $\Rightarrow$ mise en place d'un nouveau programme et retour en Usinage Automatique

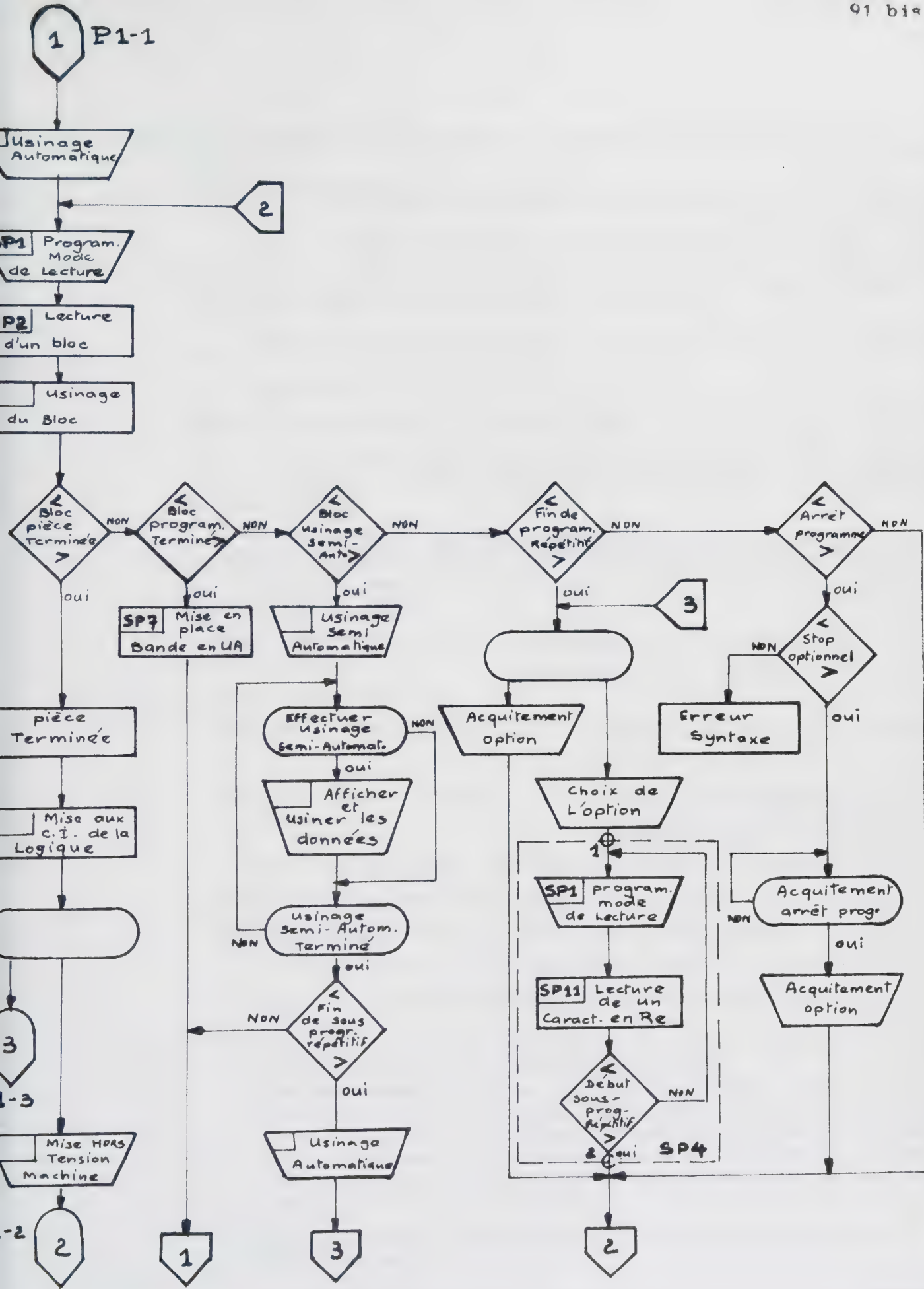
< Rupture bloc semi-automatique > $\Rightarrow$ usinage semi-automatique
puis retour en usinage automatique.

< Rupture bloc de fin de programme répétitif > $\Rightarrow$

a) si l'on choisit l'option, revenir en retour programme codé à < Rupture de début de sous-programme répétitif > et recommencer la lecture et l'usinage du sous-programme répétitif.

b) si l'on ne choisit pas l'option, continuer la lecture et l'usinage, de bloc en bloc.

.../...



Ordinogramme global des ruptures de séquences en fin d'usinage d'un bloc

Ordinogramme P3

5-3- Développement de la phase de lecture de un Bloc en usinage automatique :

5-3-1- Ordinogramme de la lecture d'un bloc (voir ordinogramme SP2 W ,page 92bis)

Lire un bloc, c'est lire successivement un ensemble de mots et mettre en mémoire tous les mots en programme du bloc, afin de les "utiliser" lors de l'usinage de ce bloc, ou de l'évolution des états globaux de la commande numérique.

Nous allons rechercher les ruptures de séquences de lecture se produisant après la lecture de un mot.

Reprenons les propriétés syntaxiques des mots :

$\langle \text{mot} \rangle ::= \langle \text{mot HORS programme} \rangle \mid \langle \text{mot EN programme} \rangle$

a) les $\langle \text{mots HORS programme} \rangle$ ne sont pas mis en mémoire dans la machine. Après la lecture d'un tel mot le système lira le mot suivant.

b) les $\langle \text{mots EN programme} \rangle$ sont mis en mémoire dans la machine. Les ruptures de séquences suivant la lecture de un mot se produisant pendant la lecture d'un bloc, sont les ruptures de bloc, c'est à dire :

$\langle \text{Rupture de bloc} \rangle ::= \langle \text{Rupture bloc optionnel} \rangle \mid$
 $\langle \text{Rupture fin de bloc} \rangle$

Soit dans un bloc :

$\langle \text{Rupture bloc optionnel} \rangle ::= \langle \text{Numéro bloc} \rangle / \langle \text{mots...mots} \rangle$
 $\downarrow$
 $\langle \text{Fin de bloc} \rangle$

$\langle \text{Rupture fin de bloc} \rangle ::= \langle \text{numéro bloc} \rangle \langle \text{mots...} \rangle$
 $\downarrow$
 $\langle \text{...mots} \rangle \langle \text{fin de bloc} \rangle$

C'est à dire après la lecture des mots suivants :

$\langle \text{Rupture bloc optionnel} \rangle \Rightarrow \langle \text{mot bloc optionnel} \rangle ::= /$

$\langle \text{Rupture fin de bloc} \rangle \Rightarrow \langle \text{mot fin de bloc} \rangle$

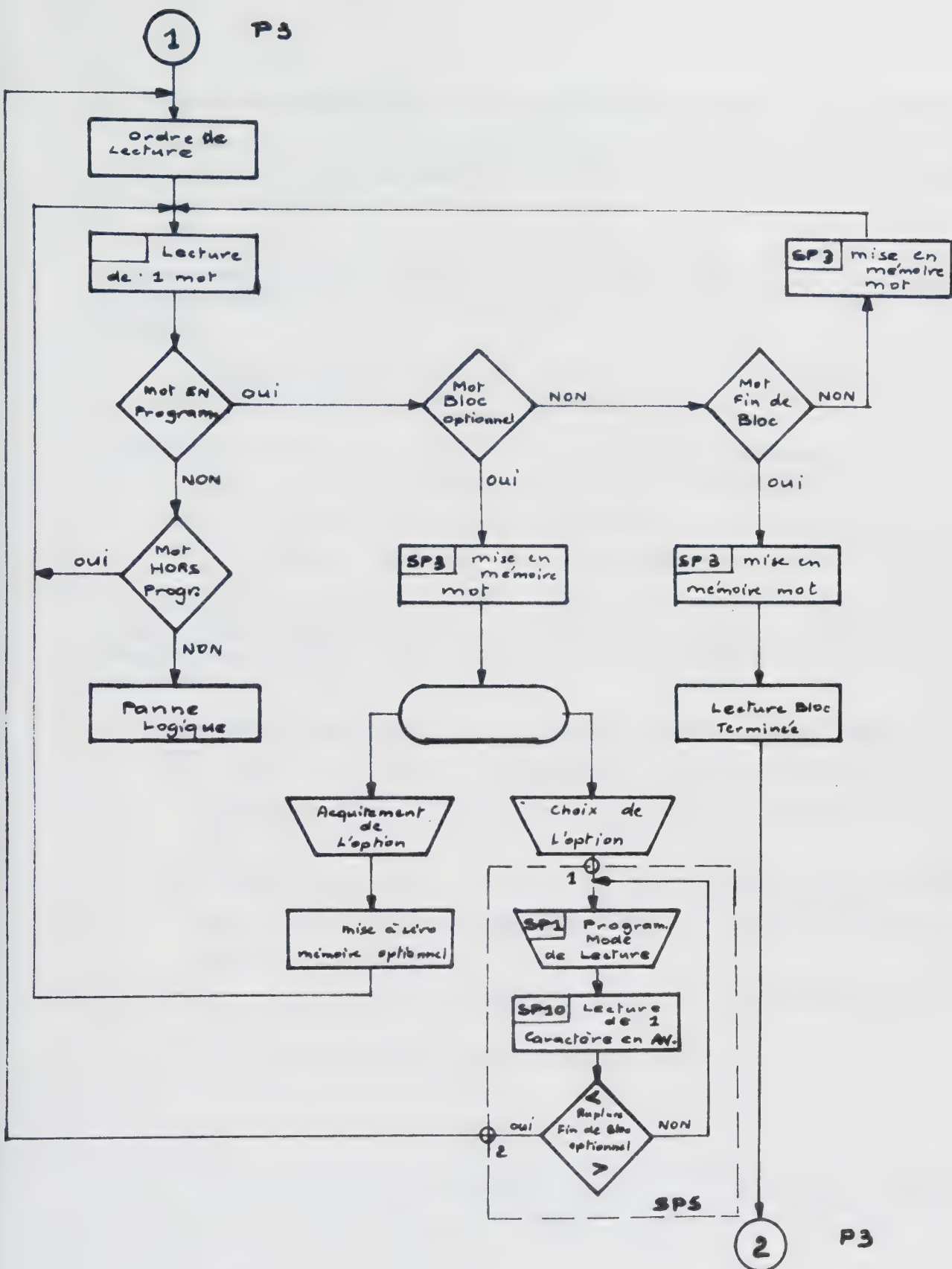
avec $\langle \text{mot fin de bloc} \rangle ::= \text{FB} \mid \text{FB} \langle \text{Fin de bloc} \rangle$

Dans le cas de $\langle \text{Rupture bloc optionnel} \rangle$ l'opérateur a le choix entre :

- usiner le bloc, c'est à dire continuer la lecture du bloc et la mise en mémoire des mots.
- ne pas usiner le bloc, c'est à dire lire tous les caractères de ce bloc sans les mettre en mémoire et lorsque la machine arrive au bloc suivant, c'est à dire au mot $\langle \text{Rupture de fin de bloc} \rangle$, de lire et de mettre en mémoire les mots de ce dernier bloc afin de les "utiliser".

Dans le cas de $\langle \text{Rupture fin de bloc} \rangle$ la lecture du bloc est terminée.

.../...



Ordinogramme provisoire développé en SP3

Sous-programme de la lecture de 1 Bloc
Ruptures de lecture dans le bloc

Ordinogramme
SP2W

Fait le 24/3/70 Par : Grossetête

5-3-2- Ordinogramme de la lecture de un caractère (Voir Ordinogramme

SP2, page 103 bis)

Reprenons l'ordinogramme précédent, de la lecture de un bloc, c'est à dire l'ordinogramme de la lecture et de la mise en mémoire d'une succession de mots.

Lire un mot, c'est lire successivement un ensemble de caractères appartenant au mot considéré et mettre en mémoire ces caractères (s'ils appartiennent à un mot EN programme), tant que le mot n'est pas terminé.

L'ensemble des ruptures de lecture des caractères est l'ensemble de rupture des mots, c'est à dire :

$\langle \text{Rupture de mot} \rangle ::= \langle \text{Rupture de mot HORS programme} \rangle \mid \langle \text{Rupture de mot EN programme} \rangle$

avec : $\langle \text{Rupture de mot HORS programme} \rangle ::= (a_i) \mid (a_i a_j \dots)$
 $\langle \text{Rupture de mot EN programme} \rangle ::= \langle \text{adresse} \rangle \mid (\mid / \mid \text{FB}$

De plus si nous introduisons les propriétés syntaxiques ci-dessous, nous en déduirons les ruptures de séquences liées aux caractères. Reprenons donc les propriétés syntaxiques étudiées au chapitre précédent.

5-3-2-1- Propriétés syntaxiques liées aux caractères :

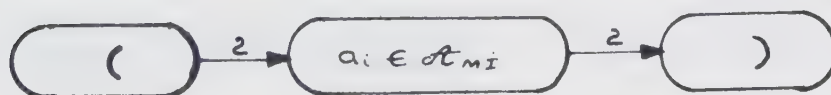
① Quel que soit le caractère $a_i \in \mathcal{A}_{MI}$ sa parité doit être correcte, c'est à dire paire en code ISO, impaire en code EIA, sauf, en code EIA, pour le caractère BL.

② Quel que soit le caractère $a_i \in \langle \text{mot HORS programme} \rangle$ c'est à dire $\langle \text{mot HORS programme} \rangle ::= (a_i) \mid (a_i a_j \dots)$

alors : $a_i \in \mathcal{A}_{MI}$

a_i est lu, mais n'est pas mis en mémoire dans la machine.

- le graphe des prédécesseurs est :



.../...

- ③ Quel que soit le caractère $a_j \notin \mathcal{A}_{M0}$
alors : a_1 est lu mais n'est pas mis en mémoire dans la machine.
- ④ Quel que soit le caractère $a_j \in \mathcal{A}_{M0}$ et appartenant à un
<mot EN programme>, il sera lu et mis en mémoire dans la machine.

5-3-2-2- Propriétés syntaxiques liées aux mots EN programme :

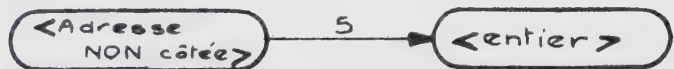
<Mot EN programme > ::= <mot ponctuation> | <mot adresse
NON cotée> | <Mot adresse cotée>

1°) Les mots adresse NON cotée s'écrivent syntaxiquement

<mot adresse NON cotée> ::= <adresse NON cotée>
<entier>

cela implique :

- ⑤ qu'un <entier> est précédé d'une <adresse non cotée>
- le graphe des prédécesseurs est :

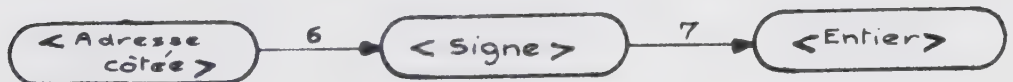


2°) Les mots adresse cotée s'écrivent syntaxiquement

<mot adresse cotée> ::= <adresse cotée> <signe> <entier>

cela implique :

- ⑥ qu'un <signe> est précédé d'une <adresse cotée>
- ⑦ qu'un <entier> est précédé d'un <signe>
- le graphe des prédécesseurs est :



.../...

3°) Les entiers s'écrivent

$\langle \text{entier} \rangle ::= \langle \text{chiffre} \rangle | \langle \text{chiffre} \rangle \langle \text{entier} \rangle$

(dans le sens de déroulement du programme dit aussi sens Avance, le chiffre de plus fort poids d'un entier est lu le premier) cela implique :

⑧ qu'un $\langle \text{chiffre} \rangle$ est précédé par le même caractère qu'un $\langle \text{entier} \rangle$

⑨ qu'un $\langle \text{chiffre} \rangle$ est précédé de un $\langle \text{chiffre} \rangle$

- le graphe des prédécesseurs est :



4°) Les mots ponctuation

$\langle \text{mot ponctuation} \rangle ::= \langle \text{Blanc} \rangle | \langle \text{optionnel} \rangle$

$\langle \text{Fin de bloc} \rangle$

$\langle \text{Blanc} \rangle ::= \text{BL} | \text{BL} \langle \text{Blanc} \rangle$

$\langle \text{Optionnel} \rangle ::= /$

$\langle \text{Fin de bloc} \rangle ::= \text{FB} | \text{FB} \langle \text{Fin de bloc} \rangle$

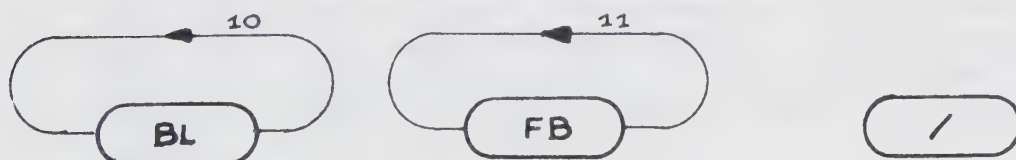
cela implique que dans un $\langle \text{mot ponctuation} \rangle$:

⑩ un caractère BL peut être précédé d'un caractère BL

⑪ un caractère FB peut être précédé d'un caractère FB

⑫ un caractère / ne peut pas être précédé d'un caractère: / .

- le graphe des prédécesseurs est :



.../...

5-3-2-3- Propriétés syntaxiques liées aux blocs :

L'enchaînement des mots dans les blocs, des blocs entre eux, conduit à étudier l'enchaînement des caractères d'un mot à ceux d'un autre mot, dans l'ensemble de tous les mots d'un programme et en particulier dans < tâche pièce > .

1°) Commençons par le début du 1er bloc de < tâche pièce > .

L'opérateur a mis en place la bande qui s'est arrêtée en position de bande prête sur :

< Rupture Avance début programme > ::=

BL BL... BL BL FB FB... FB% < entier >
 dans le sens Avance du programme codé .

Propriétés :

- ⑬ le premier caractère de < tâche pièce > lu en usinage automatique est le caractère < chiffre > qui suit le caractère %, lu en mise en place de bande dans le sens avance.

Remarque : Si l'alphabet machine-outil $\mathcal{A}_{MO}$ ne contient pas le caractère %, nous définirons différemment < Rupture Avance programme de début programme >

< Rupture Av. Début programme > ::=

BL BL... BL FB FB... FB < Numéro de bloc >
 ↓

Propriété :

- ⑭ le premier caractère de < tâche pièce > lu en usinage automatique sera alors :

le caractère qui suit le caractère FB lu en mise en place bande dans le sens avance; c'est à dire :

- FB si < Fin de bloc > ::= FB < Fin de bloc >

- N ⊕ : ● % si < Fin de bloc > ::= FB

2°) Continuons par l'étude d'un bloc en général et recherchons les propriétés d'enchaînement du dernier caractère d'un mot au premier caractère du mot suivant, quel que soit ce mot appartenant au bloc. Si les propriétés que nous allons rechercher sont vérifiées le mot précédent sera terminé et l'on pourra le mettre en mémoire des mots.

.../...

a) Reprenons les résultats de l'étude syntaxique des blocs, nous avons :

$\langle \text{Bloc} \rangle ::= \langle \text{Bloc optionnel} \rangle | \langle \text{Bloc NON optionnel} \rangle$

avec: $\left[\begin{array}{l} \langle \text{Bloc optionnel} \rangle ::= \langle \text{Fin de bloc} \rangle \langle \text{Numéro de bloc} \rangle \langle \text{mots... mots} \rangle \langle \text{Fin de bloc} \rangle \\ \langle \text{Bloc NON optionnel} \rangle ::= \langle \text{Fin de bloc} \rangle \langle \text{Numéro de bloc} \rangle \langle \text{mots... mots} \rangle \langle \text{Fin de bloc} \rangle \end{array} \right.$

avec $\langle \text{mots... mots} \rangle$, l'ensemble des $\langle \text{mots EN programme} \rangle$ et, ou des $\langle \text{mots HORS programme} \rangle$; mots : $\langle \text{Numéro de bloc} \rangle$; $\langle \text{optionnel} \rangle$, $\langle \text{fin de bloc} \rangle$, exclus.

Propriétés :

- ⑮ le caractère $\langle \text{adresse de numéro de bloc} \rangle ::= N | : | \%$ est précédé du caractère FB
- le graphe des prédécesseurs est :



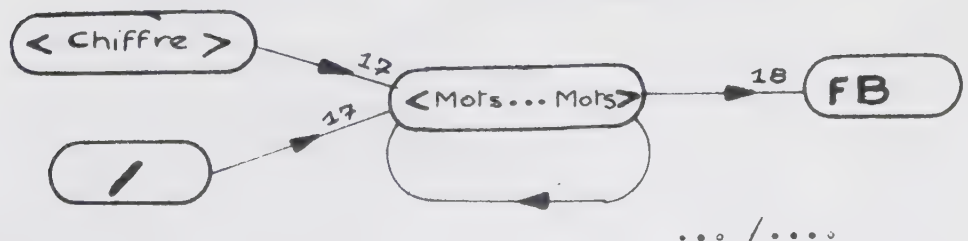
- ⑯ le caractère / est précédé du caractère $\langle \text{chiffre} \rangle$
- le graphe des prédécesseurs est :



- ⑰ le premier caractère de $\langle \text{mots... mots} \rangle$ est précédé soit du caractère $\langle \text{chiffre} \rangle$, soit du caractère /.

- ⑱ le dernier caractère de $\langle \text{mots... mots} \rangle$ est suivi du caractère FB.

- le graphe des prédécesseurs est :



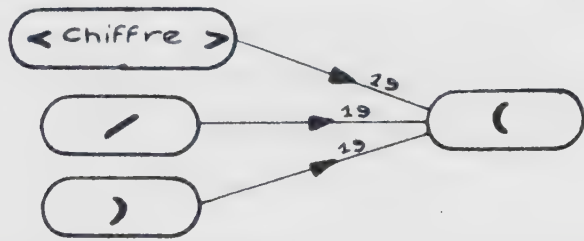
b) Etudions plus particulièrement la syntaxique de l'ensemble : <mots... mots> ci-dessus.

<mots...mots> ::= <mot HORS programme> | <mot* EN programme>
<mot HORS programme> <mots... mots> |
<mot* EN programme> <mots... mots> |
<pas de mots>

avec <mot* EN programme> : l'ensemble des mots en programme de la machine, mots <numéro de bloc>, <optionnel>, <fin de bloc> exclus.

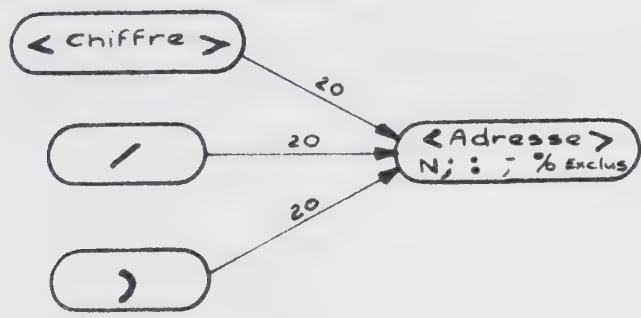
Propriétés :

- ①9 le caractère (, est précédé soit du caractère <chiffre>, soit du caractère /, soit du caractère :).
- le graphe des prédécesseurs est :



- ②0 les caractères < adresse > ::= < adresse cotée > | < adresse* non cotée >
- avec < adresse* non cotée > l'ensemble des adresses NON cotée de la machine, caractère : < adresse numéro de bloc > exclus, sont précédés soit du caractère <chiffre>, soit du caractère <optionnel> ::= /, soit du caractère] .

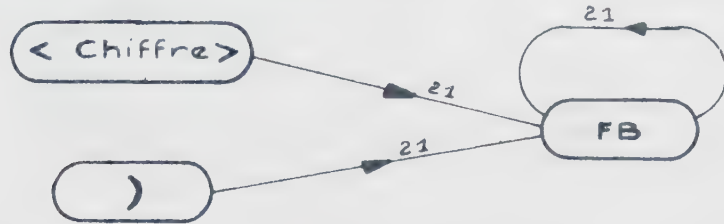
- le graphe des prédécesseurs est :



.../...

- (21) le premier caractère de $\langle \text{Fin de bloc} \rangle ::= \text{FB} | \text{FB} \langle \text{Fin de bloc} \rangle$ est précédé soit d'un caractère $\langle \text{chiffre} \rangle$, soit d'un caractère $\rangle$.

- le graphe des prédécesseurs est :



3°) Poursuivons par l'étude de l'enchaînement des blocs :

Reprenons la syntaxique d'un bloc :

$\langle \text{Bloc} \rangle ::= \langle \text{Bloc optionnel} \rangle | \langle \text{Bloc NON optionnel} \rangle$

avec :

$\langle \text{Bloc optionnel} \rangle ::= \langle \text{Fin de bloc} \rangle$

$\langle \text{Numéro bloc} \rangle / \langle \text{mots... mots} \rangle$

$\langle \text{Fin de bloc} \rangle$

$\langle \text{Bloc NON optionnel} \rangle ::= \langle \text{Fin de bloc} \rangle$

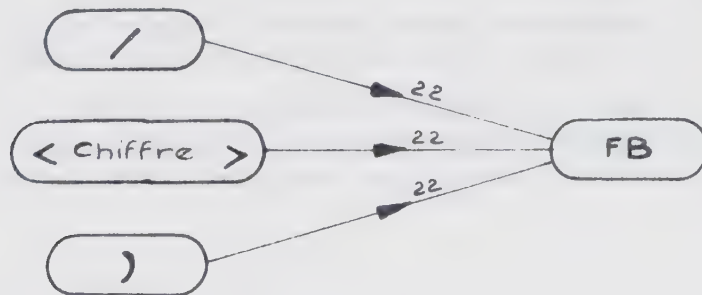
$\langle \text{Numéro de bloc} \rangle \langle \text{mots... mots} \rangle$

$\langle \text{Fin de bloc} \rangle$

$\langle \text{Fin de bloc} \rangle ::= \text{FB} | \text{FB} \langle \text{Fin de bloc} \rangle$

Propriétés :

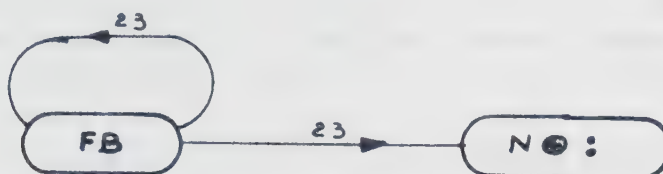
- (22) le premier caractère FB d'un bloc est le caractère de rupture de fin de bloc, il est précédé soit du caractère $/$, soit du caractère $\rangle$, soit du caractère $\langle \text{chiffre} \rangle$



.../...

- ②③ lors de la lecture d'un bloc suivant un bloc, le premier caractère lu sera :

- FB si <Fin de bloc> ::= FB <Fin de bloc>
 - <Numéro de bloc> ::= N | ; , si <Fin de bloc> ::= FB



4°) Etudions l'enchaînement de la lecture des caractères à la fin du dernier bloc d'un programme.

En fin de programme, nous avons les deux formes syntaxiques suivantes possibles :

<Bloc pièce terminée> < Blanc> |
 <Bloc programme terminé> < Blanc>

Nous avons remarqué précédemment que < Bloc pièce terminée> et < bloc programme terminé> sont des ruptures de fin de tâche pièce et que la machine n'exécute pas les informations se trouvant après le mot < Fin de bloc>

Propriétés :

- ②④ Un caractère BL ne peut être en programme.

5°) Comparons la lecture du bloc de début de tâche pièce à la lecture d'un bloc situé au milieu du programme.

Pour ce faire, nous allons reprendre les propriétés

- ①③, ①④ et ②③ que nous rappellerons.

a) Si la machine comporte le caractère % et que l'on ait fixé la < rupture Avance début programme> sur ce caractère, nous avons les propriétés suivantes :

- ①③ : le premier caractère de < tâche pièce> lu en usinage automatique est le caractère < chiffre> qui suit %, lu en mise en place de bande, dans le sens avance.

.../...

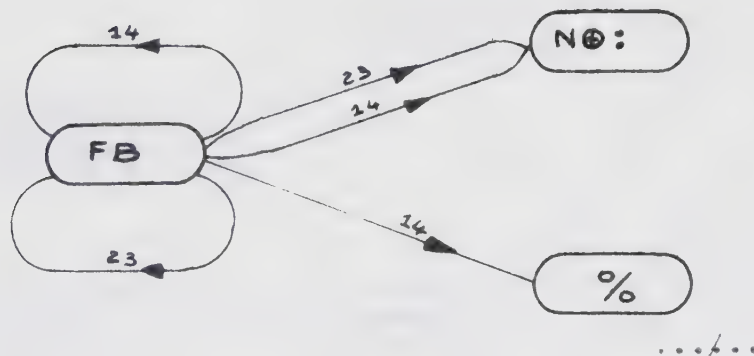
- (23) : le premier caractère lu lors de la lecture d'un bloc suivant un bloc est soit :
- FB si <Fin de bloc> ::= FB <Fin de bloc>
 <Numéro de bloc> ::= N | : si <Fin de bloc> ::= FB

b) Si la machine ne comporte pas le caractère %, ou que l'on n'a pas fini la <rupture Avance début programme> sur ce caractère, mais sur FB, nous avons les propriétés suivantes :

- (14) : le premier caractère de tâche pièce lu en usinage automatique est précédé du caractère FB lu en mise en place bande dans le sens avance, c'est à dire :
- FB si <Fin de bloc> ::= FB <Fin de bloc>
 - $N \oplus : \oplus \% \text{ si } \langle \text{Fin de bloc} \rangle ::= \text{FB}$
- (23) : Voir la propriété (23) ci-dessus que nous ne rappellerons pas une troisième fois.

c) Conclusion : Nous choisirons la < Rupture avance début programme > sur le premier caractère FB, suivant un caractère BL dans le sens d'Avance, en mise en place bande, afin de rendre identiques les enchainements de lecture des blocs.

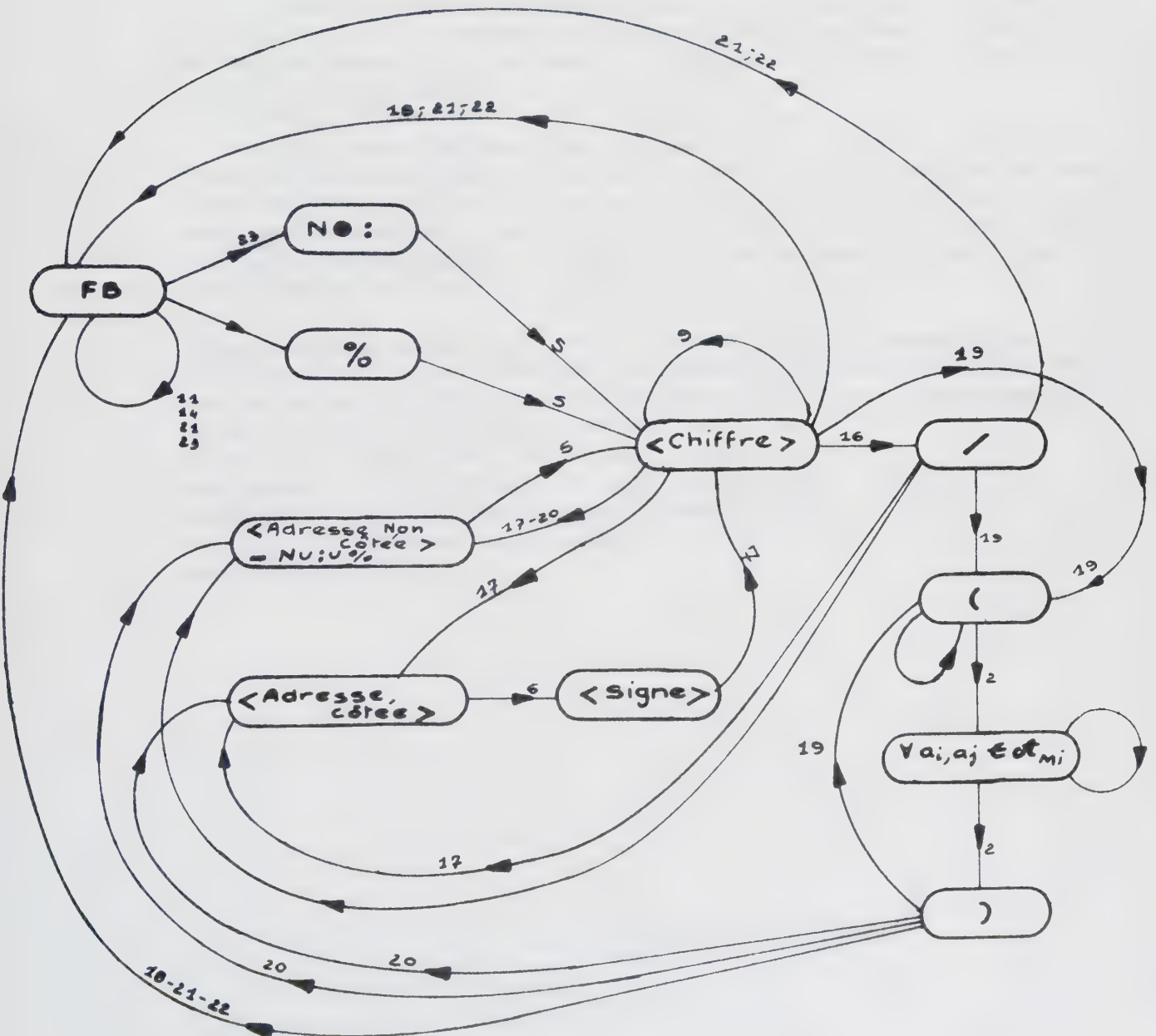
Le graphe des prédécesseurs est alors :



5-3-2-4- Graphe des prédécesseurs adjacents :

Reprenons tous les graphes des prédécesseurs que nous avons établis dans ce paragraphe au fur et à mesure de l'étude des propriétés syntaxiques.

Après avoir fusionné les sommets identiques et remplacé l'ensemble des arcs reliant deux sommets par le chemin équivalent, nous obtenons le graphe des prédécesseurs adjacents à un caractère donné lu, appartenant à un sommet déterminé.



5-3-2-5- Ordinogramme :

La reprise de l'ensemble des propriétés d'enchaînement des caractères conduit à l'ordinogramme ci-contre.

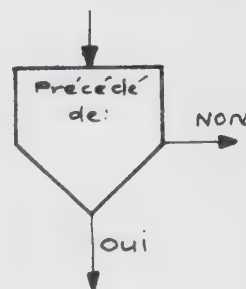
Nous avons classé les caractères lus dans l'ordre des propriétés syntaxiques des caractères et récapitulé les caractères qui doivent les précéder. Dans le cas où le caractère qui les précède (lu précédemment), n'est pas celui prévu, nous dirons qu'il y a une erreur de syntaxe de caractère.

L'ensemble des enchaînements des caractères de mot à mot conduit, si le mot précédent est EN programme à le mettre en mémoire des mots.

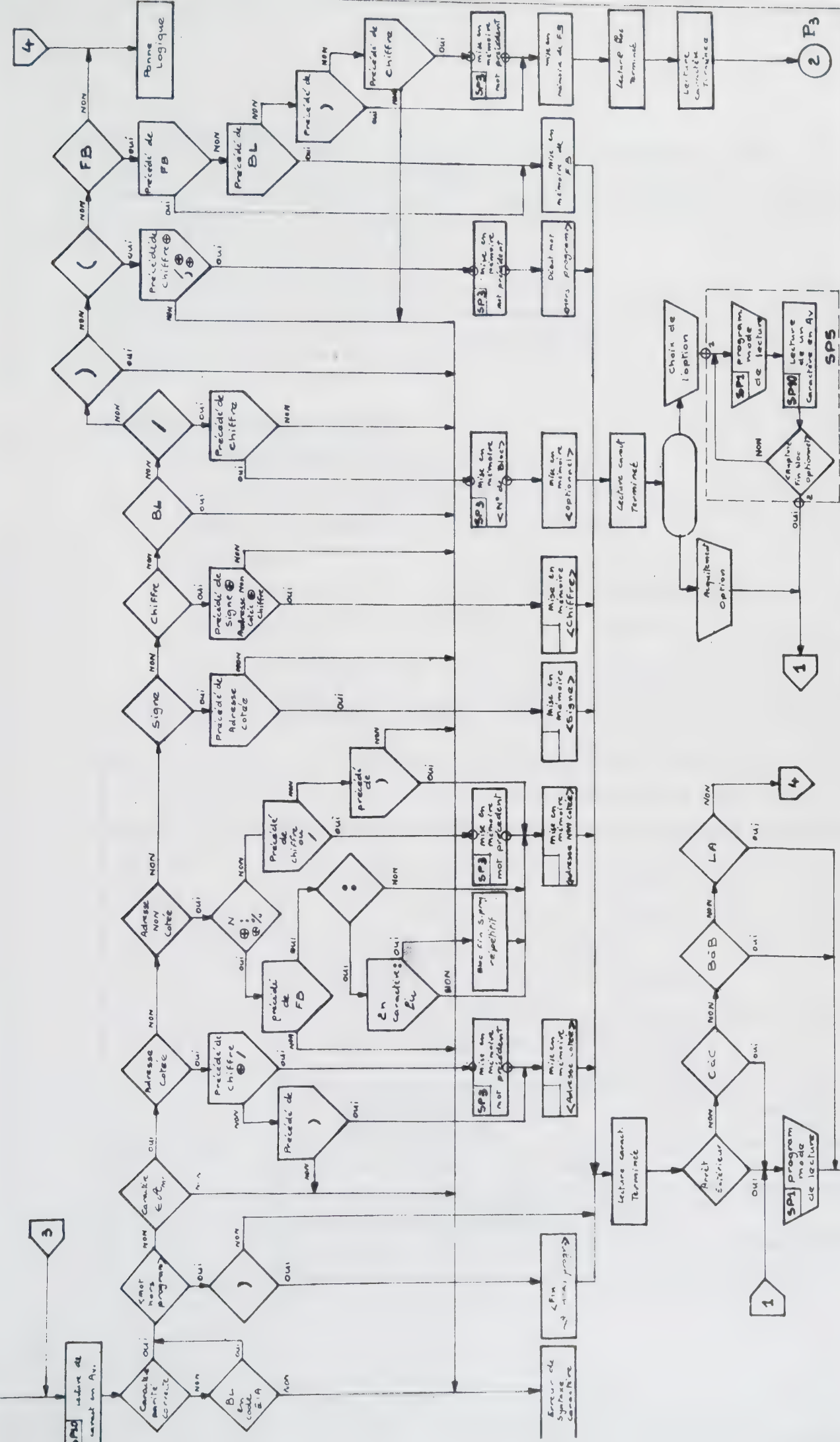
L'ensemble des enchaînements des caractères d'un mot si le mot auquel ils appartiennent est EN programme conduit à mettre ce caractère en mémoire afin de connaître l'ensemble des caractères du mot en cours de lecture.

Nous remarquerons que la commande, sait que la lecture d'un mot est terminée lorsque l'on détecte le premier caractère appartenant au mot suivant (ce ne serait pas le cas si le programme codé machine-outil était à bloc fixe, à mot fixe ou à bloc à mot adresse à format fixe, ou encore à bloc à mot adresse à format variable à rupture par tabulation).

Nota : Nous avons introduit lors de l'établissement de l'ordinogramme ci-contre un bloc diagramme particulier n'appartenant pas à la collection que nous avons détaillée au paragraphe 5-1. C'est le bloc ci-contre, qui en fait, est un bloc de décision. Il permet de préciser quels doivent être les caractères qui précèdent un caractère connu venant d'être lu. Ces caractères sont tous des prédécesseurs adjacents.



.../...



5-4- Les ordinogrammes de l'unité d'entrée à réaliser (M01)

Après avoir établi les ordinogrammes de l'unité d'entrée d'une commande numérique de machine-outil comportant toutes les propositions du langage I.S.O., nous allons établir les ordinogrammes de la machine à réaliser (notée M01).

Cette machine ne comportant pas tous les caractères, ni tous les mots du code I.S.O., nous serons amenés à supprimer les phases d'évolutions impliquées par les caractères et les mots absents. Nous obtiendrons des sous-ordinogrammes de la machine à programmation générale I.S.O.

5-4-1- Syntaxique de M01 :

1°) L'alphabet de M01 est :

- <Adresse NON cotée> ::= N|G|F|S|T|M
- <Adresse cotée> ::= I|J|X|Y|Z
- <Signe> ::= +|-
- <Chiffre> ::= 0|1|2|3|4|5|6|7|8|9
- <caractères auxiliaires> ::= BL|FB

Toutes les propriétés syntaxiques impliquées par les caractères n'appartenant pas à la machine-outil M01 ne seront pas des propriétés syntaxiques de M01, classons les par caractère.

Caractères exclus	Propositions syntaxiques modifiées
U V W P Q R	<Mot de Translation> ramenés à 3 (X Y Z)
A B C D E	<Mot de Rotation> exclus
K	Paramètre auxiliaire contournage ramené à 2 (I Y)
%	Déjà éliminé dans l'ordinogramme
:	<Sous-programme répétitif> exclus La machine est toujours en <sous-programme NON répétitif>
/	<Bloc optionnel> exclus La machine est toujours en Bloc NON optionnel
(et)	<Mot HORS programme> exclus La machine est toujours en <Mot EN programme>

2°) les mots de M01 :

Nous venons de déterminer que ce sont tous des < mot EN programme > ; que les mots d'adresse : U|V|W|P|Q|R|A|B|C|D|E|K sont éliminés, ainsi que les mots < optionnel > , < Numéro bloc répétitif > .

Il nous reste les < mots fonctions préparatoires > et les < mots fonctions auxiliaires > pour lesquels l'on se reportera à la planche 4-2. (Rappelons que tous les mots encadrés appartiennent au langage de M01).

3°) Conclusions :

Nous ne récrivons pas la syntaxique complète des blocs et des programmes de la machine M01, les éliminations des propriétés syntaxiques $P_i \notin M01$ s'effectuant simplement.

Par contre il est nécessaire d'établir les ordinogrammes de la machine M01, en éliminant les phases d'évolution qui ne se produiront jamais du fait des propositions syntaxiques $P_i \notin M01$.

5-4-2- Méthode d'élimination des phases d'évolutions :

1°) Principe :

Considérons une proposition syntaxique P_i telle que :

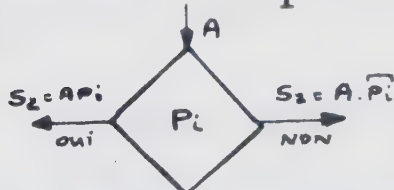
$P_i \in M0$ (Machine au langage complet ISO)

$P_i \notin M01$ (Machine à réaliser $M01 \subset M0$)

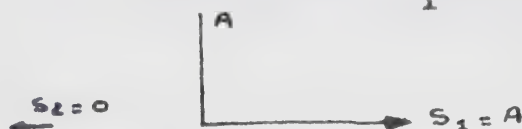
Dans la machine (M0) initiale, P_i implique que l'on a des blocs de < décision P_i >, d'état P_i atteint ou NON atteint.

Dans la machine (M01), les blocs de < décision P_i > seront toujours à l'état NON atteint, l'on ne décrira jamais le segment OUI. L'on pourra alors représenter les blocs de décision P_i de la manière suivante :

Machine M0 ; $P_i \in M0$



Machine M01 ; $P_i \notin M01$



La variable booléenne associée à P_i sera dans la machine M01 telle que :

$$\begin{aligned} \overline{P_i} &= 1 \\ P_i &= 0 \end{aligned}$$

Reportons cela dans l'équation d'autorisation d'évolution d'une phase comportant P_i ou $\overline{P_i}$. Tout minterm comportant P_i sera supprimé, c'est à dire que l'on éliminera tous les trajets i issus de (OUI) P_i (évidemment jusqu'au point de raccordement d'un trajet ne comportant pas d'état P_i).

Toute phase d'évolution, telle que son autorisation d'évolution n'est jamais vraie (P_i en facteur dans l'équation), ne se produira jamais, nous pourrions donc l'éliminer.

Toute phase d'évolution ne se produisant jamais implique que son compte rendu ne sera jamais VRAI et ainsi de proche en proche nous éliminerons toutes les phases d'évolutions impliquées dirons-nous par "uniquement" P_i .

2°) Règles d'élimination :

En résumé, obtenir l'ordinogramme de la sous-machine M01, reviendra à éliminer dans l'ordre :

- ① Tous les blocs de décisions correspondants aux états $P_i \notin$ M01 en reliant l'entrée à la sortie NON par un segment.
- ② Tous les trajets orientés issus de "uniquement" OUI P_i .
- ③ Toutes les phases d'évolutions qui ne sont impliquées par aucun trajet, et ainsi de proche en proche jusqu'à ce que l'on rencontre un trajet qui n'aie pas été éliminé.

5-4-3- Etablissement des ordinogrammes de la machine M01 :

L'on se reportera aux planches jointes (n° 5-1 à n° 5-9 inclus), précisant les ordinogrammes de la machine M01.

Pour les établir nous reprendrons les ordinogrammes de la machine M0, établis au chapitre 5 et nous réduirons les blocs de décision et les phases d'évolutions impliquées par l'état OUI des propositions $P_i \notin$ M01.

1°) Ordinogramme P_1 (M01) (voir planche n° 5-1)

Tous les blocs de décision appartiennent à M01, l'ordinogramme ne sera pas modifié.

2°) Ordinogramme P_2 (M01) (voir planche n° 5-2)

Seul l'état < Fin de programme répétitif > $\notin$ M01, car le caractère : $\notin$ M01.

Réduisons les blocs de décision < Fin de programme répétitif > que nous noterons afin de simplifier < F P R_i >. Nous allons détailler dans ce cas le processus d'élimination des phases impliquées uniquement par : OUI < F P R >.

Rappelons l'assignation informationnelle :

$$\text{OUI} < \text{FPR} > \Leftrightarrow (\text{FPR}) \quad ; \quad \text{NON} < \text{FPR}_1 > \Leftrightarrow \overline{(\text{FPR})}$$

Les équations d'Autorisations d'évolution impliquées par OUI < FPR > s'écrivent en posant :

$$\begin{aligned} \text{FPR}_1 &= \text{CRd}_1 \cdot \overline{(\text{BPT})} \cdot \overline{(\text{B Prog T})} \cdot \overline{(\text{B.UGA})} \cdot (\text{FPR}) \\ \text{FPR}_2 &= \text{CRg}_2 \cdot (\text{FPR}) \\ \text{FPR}_3 &= \text{CRi}_1 \cdot (\text{FPR}) \end{aligned}$$

Dans la machine M01 nous avons $\overline{\text{FPR}} \Rightarrow \text{FPR}_1 \text{ état } 0$
 $\text{FPR}_2 \text{ état } 0$
 $\text{FPR}_3 \text{ état } 0$
 $\overline{\text{FPR}_3} \text{ état } 1$

Reportons dans les équations des autorisations d'évolutions impliquées, il vient :

$$\begin{aligned} \text{Aj}_2 &= (\text{FPR}_2) \cdot (\text{BPUA}) \equiv 0 \Rightarrow \text{CRj}_2 \equiv 0 \Rightarrow \boxed{3} = 0 \\ \text{Af}_2 &= \left[(\text{FPR}_1) + \boxed{3} \right] \cdot (\text{BP Acq Opt}) \equiv 0 \Rightarrow \text{CRf}_2 \equiv 0 \\ \text{Ag}_3 &= \left[(\text{FPR}_1) + \boxed{3} \right] \cdot (\text{BP choix opt}) \equiv 0 \Rightarrow \text{CRg}_3 \equiv 0 \\ \boxed{2} &= (\text{FPR}_3) + \text{CRg}_3 \equiv 0 \end{aligned}$$

Il nous reste à traiter le sous-programme répétitif. Il se traite directement, si l'on remarque que tous sous-programme répétitif commence par une 1ère évolution.

Pour la 1ère évolution (variables indexées par 1) nous pouvons écrire :

$$\text{Ag}_3^1 = \text{CRg}_3^1 \equiv 0 \longrightarrow \text{CRh}_1^1 \equiv 0 \longrightarrow \text{Ai}_1^1 \equiv 0 \longrightarrow \text{CRi}_1^1 \equiv 0$$

Pour la 2ème évolution (variables indexées par 2, nous écrivons :

$$\text{Ag}_1^2 = \text{CRi}_1^1 \cdot \overline{(\text{FPR}_3)} = 0$$

En conclusion, la réduction des blocs de décision de Fin de programme répétitif élimine les phases d'évolution suivantes :

Ej₂ et le trajet d'origine CRj₂ ($\boxed{3}$)
 Ef₂, Eg₃, le sous-programme SP₄.

3°) Ordinogramme SP2 (M01) (voir planche 5-4)

Les blocs de décisions suivants correspondent à des états $P_i \notin M01$.

<Mot HORS programme>, en effet les caractères (et) n'appartiennent pas à l'alphabet de M01.

<Etat :>, le caractère : ~~∉~~ $\notin M01$

<Etat />, le caractère / ~~∉~~ $\notin M01$. Nous remarquerons que SP5 ~~∉~~ $\notin M01$.

<Etat)>, le caractère) ~~∉~~ $\notin M01$

<Etat (>, le caractère (~~∉~~ $\notin M01$

<précédé de)>, le caractère) ~~∉~~ $\notin M01$

Après avoir éliminé les phases d'évolutions impliquées "uniquement" par les états OUI de ces blocs de décision, nous sommes conduits à l'ordinogramme SP2 (M01).

4°) Les autres ordinogrammes :

Après avoir appliqué ces méthodes aux développements de chaque phase d'évolution, nous nous sommes arrêtés à l'instant où, pour détailler plus, il était nécessaire de connaître les fonctions logiques binaires qui doivent réaliser ces évolutions.

Nous avons été conduits à l'établissement des ordinogrammes suivants :

- a) la programmation manuelle en usinage automatique (ordinogramme SP1/M01; Planche 5-3)
- b) la mise en mémoire des mots (ordinogramme SP3/M01 ; Planche 5-5)
- c) la mise en place de la bande en préparation lecteur et en usinage automatique (ordinogramme SP8 et SP9/M01 ; planche 5-7)
- d) la lecture de un caractère en Avance, en Retour sur la bande programme (ordinogramme SP 10, SP 11 ; planche 5-8)
- e) les arrêts extérieurs et acquittements de ces arrêts (ordinogramme SP 12 M01 ; planche 5-9). L'on remarquera que c'est un ordinogramme indépendant arrêtant toute phase d'évolution de la machine si l'on est en arrêt extérieur.

6 - UNITES FONCTIONNELLES

(Recherche des boucles essentielles)

pages :

<u>6-1- Introduction à la synthèse générale</u>	111
6-1-1- Phases d'évolutions globales	111
6-1-2- Etats de la synthèse générale machine	113
6-1-3- Graphe des évolutions globales	114
6-1-4- Conclusions	115
<u>6-2- Introduction à la programmation manuelle, et à la lecture de un caractère</u>	116
6-2-1- La lecture en usinage semi-automatique	116
6-2-2- La lecture en usinage automatique	117
6-2-3- La lecture en mise en place bande	118
6-2-4- Conclusions	121
<u>6-3- Conclusions :</u>	122
6-3-1- Résumé des propriétés essentielles des différentes unités	123
6-3-2- Introduction à l'étude des unités	127

INTRODUCTION :

Nous venons de définir les ordinogrammes de la machine à réaliser (M01), l'énoncé des propriétés générales de la machine, est posé.

Il n'est pas prudent de se lancer au hasard dans l'étude de petites machines logiques plus ou moins évidentes, ni d'avoir la prétention d'effectuer une synthèse formelle de toute la machine, en déterminant son graphe complet (aux évolutions d'états internes) en détaillant tous les états possibles et tous les chemins possibles.

Nous allons déterminer des machines dites fonctionnelles, c'est à dire des unités de logique qui réalisent un sous-ensemble déterminé des phases d'évolutions de même application sémantique pour tous les ordinogrammes (l'on établit une partition des phases d'évolutions de même application sémantique).

Pour déterminer ces machines nous commencerons par dégager les boucles essentielles de l'ensemble des ordinogrammes reliant les états essentiels de la machine. Nous établirons des graphes particuliers qui ne seront que des représentations très simples des ordinogrammes.

A un sommet correspond un état essentiel de la machine. Dans cet état la machine décrit un ensemble de phases d'évolutions de l'ordinogramme.

A un chemin reliant deux sommets, correspond un trajet ou un ensemble trajets de l'ordinogramme reliant les sous-ensembles de phases d'évolutions contenues dans les sommets.

L'étude que nous effectuerons sera rapide, pourra paraître imprécise car, plutôt que d'établir le graphe général correspondant aux ordinogrammes, nous sommes partis des états essentiels de la machine imposés par la sémantique de celle-ci.

Nous commencerons par établir les phases d'évolutions globales et le graphe aux évolutions globales, puis les états de synthèse de la machine pour finalement arriver au graphe des évolutions globales en fonction des états de synthèse. Cette étude nous permettra de dégager deux unités fonctionnelles :

- L'Unité de synthèse générale, non étudiée dans cette Thèse ;
- L'Unité de programmation manuelle des états globaux.

Ceci établi, nous développerons à nouveau les sommets du graphe des évolutions globales en fonction des états de synthèse. Nous serons conduits aux graphes de la programmation manuelle en fonction des états de lecture de un caractère. Cette étude nous permettra de dégager deux unités fonctionnelles.

- L'Unité de programmation manuelle de la lecture
- L'Unité de lecture de un caractère.

Il ne nous restera plus que la machine autonome de l'arrêt extérieur (Ordinogramme SP 12 [Planche 5 -9]).

.../...

6-1- Introduction à la synthèse générale de la machine :

Il n'est pas pensable, surtout sans l'aide de calculateur, de déterminer le graphe complet de la machine et de déterminer les noeuds et les boucles essentielles. Nous travaillerons donc de façon plus intuitive, en raisonnant sur les ordinogrammes.

6-1-1- Phases d'évolutions globales :

Considérons les ordinogrammes globaux de la machine, c'est à dire les ordinogrammes P_1 (MO1) et P_3 (MO1) [Planches 5-1 et 5-2]. Nous en dégageons les phases globales d'évolutions et les enchainements entre ces phases, en fonction des ruptures de séquences, et des évolutions intermédiaires.

Plutôt que de citer des phases globales, représentons le tout sur le graphe très simplifié ci-dessous qui n'est rien d'autre qu'une représentation très synthétique des ordinogrammes.

- . Les sommets sont repérés par un numéro renvoyant au nom de la phase globale ;
- . Les chemins représentent des phases d'évolutions intermédiaires, des actions manuelles et des < états > qui permettent d'aller d'une phase à l'autre.

La symbolisation utilisée pour les états ou actions manuelles situées sur les chemins est :

BP PL : action manuelle sur le Poussoir, préparation lecteur
 BP MPB : action manuelle sur le poussoir, mise en place bande
 BP BP : action manuelle sur le poussoir, bande prête
 BP UA : action manuelle sur le poussoir, usinage automatique
 BP USA : action manuelle sur le poussoir, usinage semi-automatique.

M MO2 : mot MO2 en mémoire (mise en mémoire en E_3 b_1)
 M M10 : " M10 " "
 M M30 : " M30 " "

- ⑧ La mise en place de la bande, appelée par la rupture de programme $\langle \text{Bloc programme terminé} \rangle ::= M30$ soit en mémoire M M30 (Phases $E_{III}e_1$)
- ⑨ Bande prête après appel du programme (phase d'évolution incluse dans $E_{III}e_1$ voir SP 7).

6-1-2- Etats de la synthèse générale de la machine :

Considérons les évolutions de l'unité d'entrée et de l'unité d'usinage pour chaque phase d'évolution globale précédente. Par phase, dégageons les boucles essentielles à l'intérieur de la phase. Les chemins incidents et émergents à chaque phase seront évidemment des chemins incidents et émergents des "sous-phases" que nous allons dégager.

1°) Etudions par exemple la phase globale d'évolution ⑥ d'usinage automatique normal.

a) Rappelons l'hypothèse que nous avons faite lors de l'établissement de l'ordinogramme P2 W. La machine lit le bloc N°n (lecture en cours : LC_6); lorsque la lecture est terminée (LT_6), l'unité d'usinage usine le bloc (usinage en cours UC_6); lorsque l'usinage est terminé (UT_6) la machine reprend le cycle ou quitte la phase d'évolution ⑥ si l'on a une rupture de programme du type (M M02, M M10, M M30)

b) Représentons cela sur un graphe qui n'est autre, encore une fois, que la reproduction très synthétique de l'ordinogramme.

- Les sommets sont repérés par UT_6 , LC_6 , LT_6 , UC_6 . Ils correspondent à des phases d'évolution des ordinogrammes.
- Les chemins représentent aussi des phases d'évolution des $\langle \text{états} \rangle$, des actions manuelles.

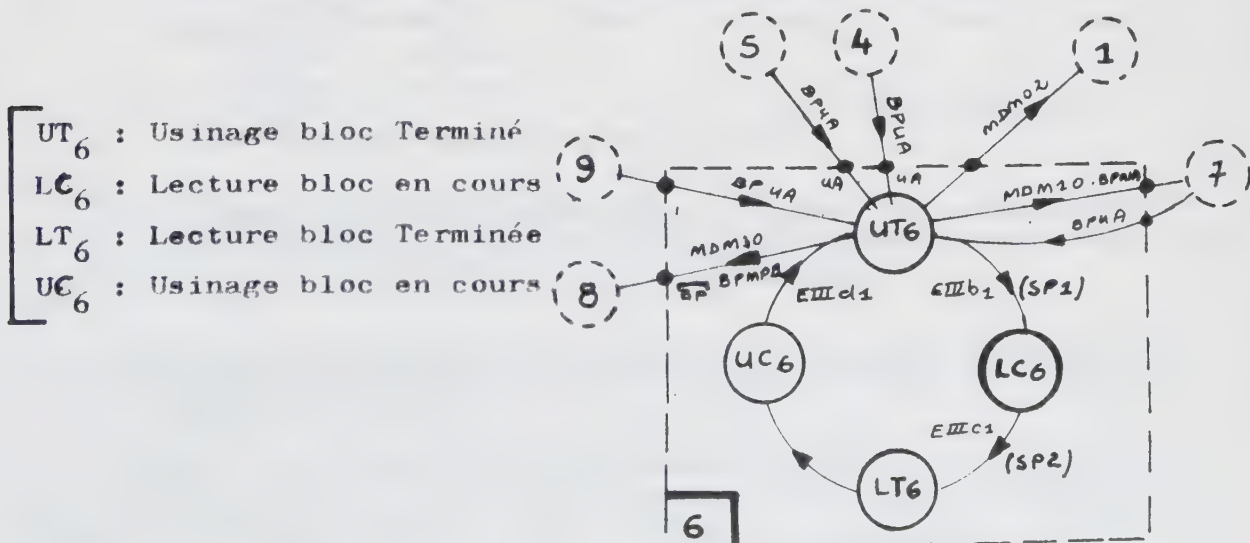


Figure 2

c) Sommet initial et final :

• Le sommet initial et le sommet final de ce cycle ne peuvent être que UT_6 (Cas exclus de la mise aux conditions initiales forcées, ou un ordre spécial de secours, afin de ne pas rester bloqué dans une phase).

• Les chemins incidents et émergents à la phase (6) Figure 1, sont des chemins incidents et émergents du sommet UT_6 .

d) Détail de quelques chemins :

Les chemins sont des suites de phases d'évolution des ordinogrammes. A titre d'exemple, afin d'éclairer la représentation du graphe, nous allons détailler quelques chemins dans leur ordre d'apparition, en commençant par la première arrivée sur le cycle (chemin⁴ → 6)

- chemin: $(4) \rightarrow (UT_6) = P_{A-1}$, $E_{III}^{a,1}$ (représenté par BP UA, impliquant l'état UA à l'arrivée dans (6)).
- chemin: $(UT_6) \rightarrow (LC_6)$
 - le premier : $E_{III}^{b,1}$ (programmation du mode de lecture en usinage automatique)
 - les suivants : si $\overline{M \ M02} \cdot \overline{M \ M30} \cdot \overline{M \ M10} \dots$
 $\boxed{2}$, $E_{III}^{b,1}$

Nous remarquerons que si les ruptures de programme ci-dessus ne sont pas dans l'état indiqué, l'on prend un chemin émergent à (6).

- chemin: $(LC_6) \rightarrow (LT_6)$: c'est l'ensemble des évolutions de l'ordinogramme SP2 (M01), de (1) SP2, à (2) SP2 (Boucles $\boxed{3}$ incluses).
- chemin: $(LT_6) \rightarrow (UC_6)$: non développé dans les ordinogrammes
- chemin: $(UT_6) \rightarrow (7)$: c'est l'ensemble des évolutions telles que l'on ait :
 $\overline{M \ M02} \cdot \overline{M \ M30} \cdot M \ M10$, $E_{III}^{e,2}$ (représenté par BP USA impliquant un état USA.)

2°) Nous ne développerons pas ici toutes les phases d'évolutions globales, l'on se reportera au graphe des évolutions globales ci-après (Figure 3 page 115 Bis)

6 -1-3- Graphe des évolutions globales en fonction des états de synthèse générale.

Après avoir étudié successivement toutes les phases globales d'évolution, leurs enchainements, les états initiaux et finaux de chaque phase, l'on est conduit au graphe ci-après

Nous avons représenté les 9 phases d'évolution globales. Sur les chemins nous avons fait figurer les actions manuelles et les évolutions essentielles des ordigrammes.

Plutôt que d'étudier une machine logique comportant les états : UT_1 , LC_1 , LT_1 , UC_1 de chaque phase n°1, nous avons simplifié en différenciant les phases d'évolution de ① à ⑨ par la programmation des états globaux.

Nous aurons deux unités fonctionnelles ; l'une élaborant les 4 états de synthèse générale (UT , LC , LT , UC), l'autre les états de programmation globale, impliqués par les actions manuelles (BP PL BP USA) en fonction des états des ruptures de séquences.

。 Précisons les états de la programmation globale stable dans chaque phase globale (sur le cycle intérieur).

phase globale	PL	MPB	BP	UA	USA	MD M02	MD M10	MD M30	phase globale	PL	MPB	BP	UA	USA	MD M02	MD M10	MD M30
1	0	0	0	0	0	0	0	0	5	0	0	1	0	1	0	0	0
2	1	1	0	0	0	0	0	0	6	0	0	1	1	0	0	0	0
3	0	0	0	0	1	0	0	0	7	0	0	1	0	1	0	1	0
4	1	1	1	0	0	0	0	0	8	0	1	0	1	0	0	0	1
									9	0	1	1	1	0	0	0	0

。 Précisons les états possibles de synthèse générale :

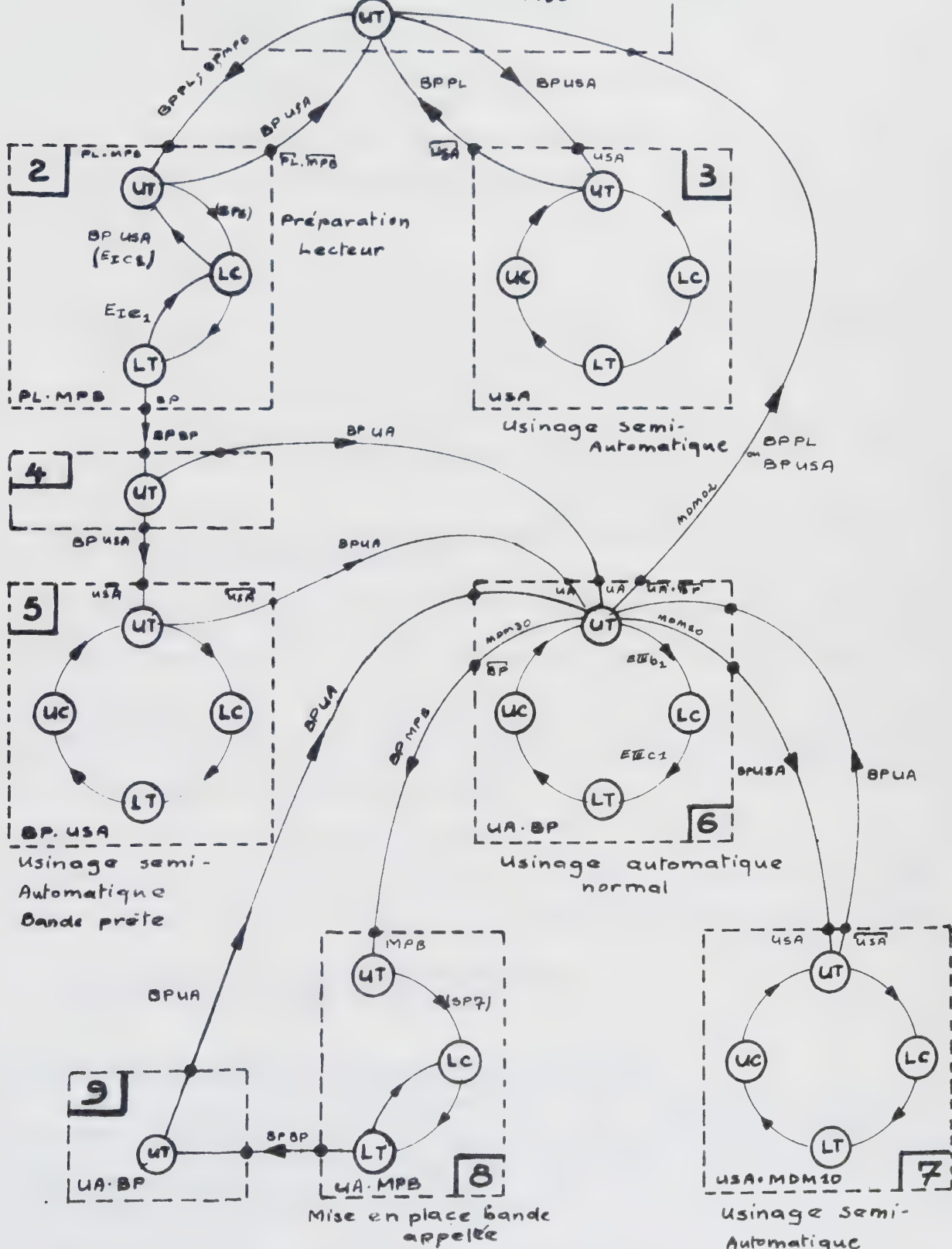
	UT	LC	LT	UC
état de transition unique	0	0	0	0
	1	0	0	0
états stables.	0	1	0	0
	0	0	1	0
	0	0	0	1

6-1-4- Conclusions :

1°) L'unité de synthèse générale :

Ce sera une machine logique à 4 états : UT , LC , LT , UC . Nous ne l'étudierons pas dans cette thèse, car pour la déterminer, il est nécessaire de connaître les signaux de l'unité d'entrée et de l'unité d'usinage.

Mise aux
Conditions
initiales



Graphes des évolutions globales
en fonction des états de synthèse

Figure 3

2°) L'unité de programmation manuelle globale :

C'est la machine logique qui en fonction :

- des actions manuelles : BP PL ; BP MPB ; BP BF ; BP UA ; BP USA .
- des états de rupture de programme : M M02, M M10, M M30
- des états de synthèse générale : UT, LC, LT, UC

élabore les états globaux précisés sur le tableau du paragraphe 6-1-3.

On remarquera que :

- Les mémoires de rupture de programme ci-dessus doivent être mises à zéro sur les chemins $\textcircled{8} \rightarrow \textcircled{6}$; $\textcircled{6} \rightarrow \textcircled{1}$ $\textcircled{7} \rightarrow \textcircled{6}$ de la figure 1 en synchronisme avec UT, alors qu'elles sont mise à N en synchronisme avec LC.

- Les changements d'états de PL, MPB, UA, USA sont synchrones par rapport à UT ainsi que la mise à $\overline{\text{BP}}$ de BP. La mise à l'état BP étant synchrone avec LT.

6-2- Introduction à la programmation manuelle de lecture et aux phases de lecture de un caractère :

Notre but, n'est pas d'étudier en détail les ordinogrammes. Nous rechercherons encore une fois à isoler des phases d'évolution de même application sémantique afin de déterminer des machines fonctionnelles.

Nous déterminerons les états essentiels de la lecture de un caractère et les boucles essentielles reliant ces états. Nous développerons donc les évolutions se produisant pendant l'état de lecture en cours LC pour les différentes phases globales.

Nous avons trois types de lecture :

- la lecture en usinage automatique normal [phase globale : $\textcircled{6}$]
- la lecture en mise en place de la bande [phases globales : $\textcircled{2}$ - $\textcircled{8}$]
- la lecture en usinage semi-automatique [phases globales : $\textcircled{3}$ - $\textcircled{5}$ - $\textcircled{7}$].

6-2-1- La lecture en usinage semi-automatique :

Ce point ne sera pas abordé dans cette thèse, les ordinogrammes n'ont d'ailleurs pas été détaillés. Indiquons simplement que la lecture se résume à un affichage direct (par rotacteurs des mots) dans les registres mémoire des mots (limités aux mots permettant l'usinage point à point).

6-2-2- La lecture en usinage automatique normal (phases globales ⑥).

Reprenons la phase ⑥ du graphe des évolutions globales et étudions les phases d'évolutions se produisant pendant cette phase, c'est à dire à l'intérieur de ⑥ et tout particulièrement à l'intérieur de Lc_6 .

Le trajet incident à Lc_6 est ① de SP2 (M01)

Le trajet émergent de Lc_6 est ③ de SP2 (M01)

La seule boucle essentielle de l'ordinogramme SP2 est la boucle ③ de SP2.

La machine peut être dans deux états essentiels de lecture de un caractère pendant le cycle ③ :

[$Lc\ aC$: lecture de un caractère en cours
 $Lc\ aT$: lecture de un caractère terminée.

La lecture de un caractère correspondant à l'ensemble des phases d'évolution de SP2, de l'évolution E_2a_1 incluses, aux évolutions E_2g_1 et E_2c_3 en cours.

L'état initial et l'état final ne peuvent être que lecture caractère terminée. En UT_6 , LT_6 , Uc_6 l'on ne peut être que en l'état $LcaT$.

En lecture en cours Lc_6 , la machine décrit le cycle précisé sur le graphe ci-après, sur lequel nous avons indiqué les chemins incidents et émergents à Lc_6 .

Nous représenterons le graphe de la façon suivante :

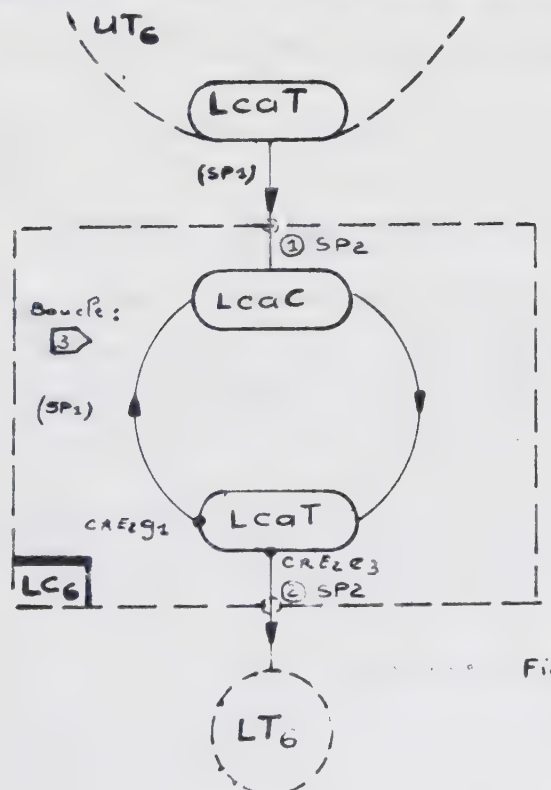


Figure 4

De chemins :

$(UT_6) \rightarrow (Lc\ aC)$: l'ensemble des évolutions du compte-rendu de l'action manuelle usinage automatique (1) de SP1 à (1) de SP2, qui fait passer UT_6 à LC_6 et ensuite fait passer à l'état lecture caractère en cours (évolution $E_2\ a_1$).

$(LcaC) \rightarrow (LcaT)$: l'ensemble des évolutions de $E_2\ a_1$ à $E_2\ e_3$ ou $E_2\ g_1$ incluses.

$(LcaT) \rightarrow (LcaC)$: la bande de programmation $\boxed{3}$ de CR $E_2\ g_1$ à $\boxed{3}$

$(LcaT) \rightarrow LT_6$: le compte-rendu d'évolution CR $E_2\ e_3$ implique l'évolution de la synthèse générale vers LT_6

6-2-3)- La lecture en mise en place de la bande :

(Phases globales (2) - (8) de la figure N°1 page *112)

Considérons ces deux phases globales et les ordinogrammes associés (essentiellement SP6-SP7 (M01) Planche 5-2 et SP8-SP9 Planche 5-8).

Nous allons à nouveau développer la phase de synthèse de lecture en cours.

1°) Avant d'établir les graphes nous allons comparer les deux phases d'évolutions (2) et (8) et relever les propriétés n'appartenant pas aux deux phases globales.

a) Les états respectifs de la programmation manuelle globale :

- Phase (2) : PL . MPB . BP . UA . USA . M M02 . M M10 . M M30
 - Phase (8) : PL . MPB . BP . UA . USA . M M02 . M M30
- (M M10 est indifférent).

b) Les chemins internes appartenant à (2) et pas à (8)

En nous reportant aux ordinogrammes, nous avons :

- le chemin $(LC_2) (UT_2)$. C'est à dire l'ensemble des phases d'évolutions issues de CR e_1 , CR e_2 , CR e_3 , CR e_4 , passant par (3) P1, puis $E_1\ c_2$ (c'est la boucle $\boxed{2}$ de P1 (M01) conduisant à la phase (3) d'usinage semi-automatique bande non prête.

- les chemins $(LT_2) \rightarrow (LC_2)$ et $(LT_8) \rightarrow (LC_8)$ sont identiques, ils correspondent à l'acquittement de début de programme (phase: $E_1 e_1$) dont on retrouvera facilement le détail sur l'ordinogramme SP6 - SP7, (Phases: $E_6 f_2, E_6 f_3, E_7 f_2, E_7 f_3$).

c) Les chemins incidents et les chemins émergents sont différents, nous ne les détaillerons pas (il suffit de se reporter au graphe des évolutions globales en fonction des états de synthèse figure 3. page 115 bis)

d) Nous allons étudier la phase (2) qui contient tous les chemins en représentant en pointillé les chemins n'appartenant pas à (8) .

Aucune ambiguïté n'étant possible quant aux chemins incidents et émergents, l'on déterminera facilement (8) à partir de (2) en changeant les chemins.

2°) Etude de la phase (2) .

Comme pour la phase (6) , en usinage normal, nous allons développer la phase d'évolution (2) .

Nous avons deux cycles de lecture. La lecture en avance bande, la lecture en Retour Bande.

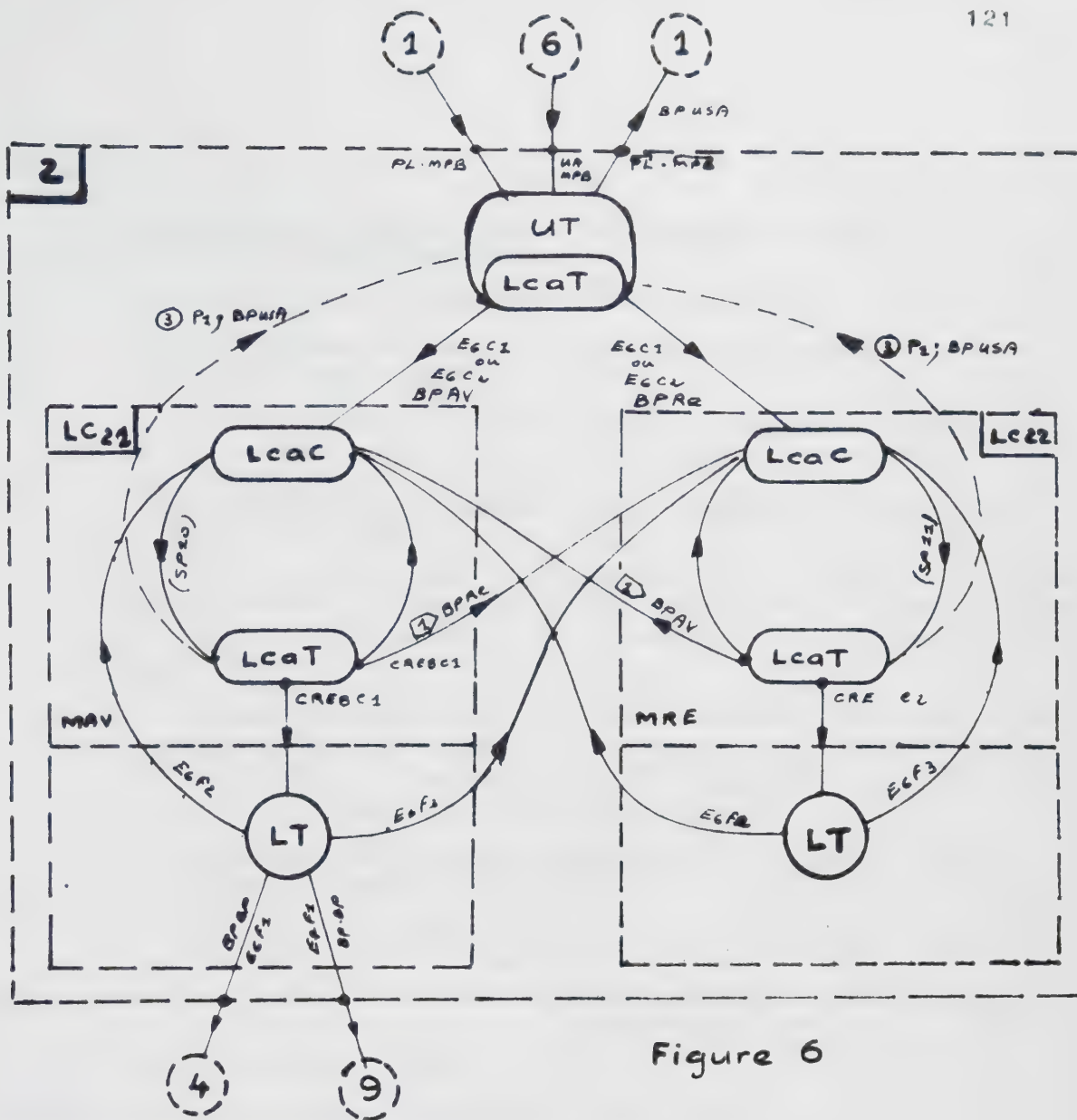
Plutôt que d'établir le graphe reprenons nos considérations générales de sémantique de la machine. Celle-ci peut être dans les 4 états suivants :

[	LC Av : Lecture en cours en Avance
	LT Av : Lecture Terminée en Avance
	LC Re : Lecture en cours en Retour
	LT Re : Lecture Terminée en Retour.

Ceci nous conduit à "dupliquer" les états LC et LT par un état Avance et un état Retour, notés: MAV et MRE.

Nous pouvons alors tracer le graphe suivant pour les évolutions à l'intérieur de la phase (2) , en faisant apparaître deux sous-graphes, notés: (21) pour Av et (22) pour MRe. Nous avons repéré les chemins de ce graphe par les symboles des trajets correspondants sur l'ordinogramme (Figure 5 page 120)

.../...



6-2-4- Conclusions :

Reprenons les graphes de l'usinage normal et la mise en phase Bande (figures 3 et 6).

Nous en déduisons deux machines fonctionnelles.

1°) L'unité fonctionnelle de lecture de un caractère d'états LcaT ou LcaC.

. Nous réaliserons une machine unique de lecture de un caractère décrivant l'ensemble des phases d'évolutions des ordinogrammes SP8 (M01) - SP9 (M01) et la partie des phases d'évolutions de SP2 (M01) entre E_2a_1 et E_2e_3 ou E_2g_1 incluses.

En effet ces ordinogrammes ne sont jamais décrits simultanément et ils seront différenciés dans les évolutions de la machine par les états de la programmation globale et en plus par MAV ou MRE dans le cas particulier de la mise en place bande.

2°) L'unité fonctionnelle de la programmation manuelle de la lecture de la bande programme.

Nous avons deux types de fonctionnement :

- la programmation manuelle en usinage automatique normal (en phase (6)).
- la programmation en mise en place bande (en phases (2) et (8)).

a) La programmation manuelle en usinage automatique normal Phase (6).

C'est l'ensemble des phases d'évolutions de l'ordinogramme SP1. C'est à dire les phases d'évolutions situées sur le chemin initial UT_6 / LcaC et sur le chemin LcaT, LcaC.

Nous remarquerons que les évolutions dans ces ordinogrammes ne peuvent se produire que si l'on est dans l'état LcaT, et que de plus les états impliqués par SP1 ne sont utilisés que dans la phase (6).

b) La programmation manuelle en mise en place de la bande (phases (2) et (8)).

C'est l'ensemble des phases d'évolution de l'ordinogramme SP6 - SP7, phases d'évolution ci-dessous exclues

	phases exclues
Mise en place bande	E_6a
Lecture de un caractère en Avance	E_6e_1, E_6e_3
Lecture de un caractère en Retour	E_6e_2, E_6e_4
Bande prête	E_6

Nous remarquerons que les évolutions dans ces phases non exclues ne peuvent se produire que si l'on est dans l'état LcaT, et que de plus aucun état impliqué par ces phases n'est utilisé hors des phases d'évolutions (2) et (8).

6-3- Conclusions :

Nous venons de dégager quatre machines fonctionnelles.

- L'unité de synthèse générale
- L'unité de programmation manuelle des états globaux
- L'unité de programmation manuelle du mode de lecture
- L'unité de lecture de un caractère.

Nous avons établi une partition dans l'ensemble des phases d'évolutions de la machine telle que dans une partie donnée l'on trouve toutes les phases d'évolution de même application sémantique.

De plus nous avons déterminé ces machines en fonction des noeuds essentiels et des boucles essentielles des ordinogrammes qui correspondent aux boucles essentielles que l'on établit sur les ordinogrammes en les déterminant.

Nous n'avons pas tracé le graphe général de l'ensemble, sans calculateur ; il est fort improbable que l'on puisse en tirer des enseignements utiles et de plus l'on a intérêt pratiquement, en logique câblée industrielle à déterminer de petites machines logiques ayant des fonctions précises plutôt qu'une seule machine.

6-3-1- Propriétés essentielles des différentes unités :

Avant d'étudier ces machines nous allons préciser pour chacune les propriétés essentielles.

1°) L'unité de synthèse générale de la machine :

Elle ne sera pas étudiée dans cette thèse. Nous avons 4 états possibles de fonctionnement UT, LC, LT, UC qui seront matérialisés par 4 signaux booléens du même nom.

	UT	LC	LT	UC
Etat de transition unique	0	0	0	0
Etat UT	1	0	0	0
Etat LC	0	1	0	0
Etat LT	0	0	1	0
Etat UC	0	0	0	1

2°) L'unité de programmation des états globaux :

Le tableau des états stables de fonctionnement dans chaque phase globale est le suivant.

Phase globale	PL	MPB	BP	UA	USA	M M02	M M10	M M30
1	0	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0
3	0	0	0	0	1	0	0	0
4	1	1	1	0	0	0	0	0
5	0	0	1	0	1	0	0	0
6	0	0	1	1	0	0	0	0
7	0	0	1	0	1	0	1	0
8	0	1	0	1	0	0	-	1
9	0	1	1	1	0	0	0	0

.../...

• Etats interdits :

Les considérations évidentes quant à la sémantique de la machine nous permettent d'établir les états interdits ci-contre.

Notre notation indique que PL à l'état 1 implique USA à l'état zéro .

	PI	MPB	BP	UA	USA	M M02	M M10	M M30
PL →	1	1	1	0	0			
MPB →					0			
BP →								
UA →	0				0			
USA →	0	0			0			

• Propriétés générales :

Les changements d'état de (PL, MPB, UA, USA) sont synchrones par rapport à l'état UT de la synthèse générale, le changement d'état de BP est synchrone par rapport à l'état LT pour la mise à un de BP et synchrone par rapport à UT pour la mise à zéro de BP.

Les états mémoires des mots M M10, M M30, M M02 apparaissent pendant la phase LC₆ (phase d'évolution E₃₁ des ordinogrammes). Les états un des mémoires de ces mots impliquent les ruptures de cycle en UT₆ : UT₆•M M02 ; UT₆•M M10 ; UT₆•(M M30) impliquent les chemins émergents de la phase globale ⑥ . Les états M M02, M M10, M M30 des mémoires ne peuvent appartenir au chemins incidents à UT₆ car nous aurions un cycle. Nous effectuerons donc la mise à l'état zéro de ces états mémoires hors de la phase ⑥ .

3°) L'unité de programmation manuelle du mode de lecture .

a) En usinage automatique normal phase ⑥

• Le Tableau des états stables en fonctionnement dans la phase ⑥, impliqué par les phases d'évolutions de SPI est le suivant :

Etats stables possibles en phase ⑥	LM	LA	C à C	B à B
Lecture manuelle caractère à caractère	1	0	1	0
Lecture manuelle bloc à bloc	1	0	0	1
Lecture automatique	0	1	0	0
Etat stable hors de la phase ⑥	1	1	0	0

• Le tableau des états interdits

est représenté ci-contre :

Notre notation indique que LM à l'état 1, implique LA à l'état zéro.

LM →		0		
LA →	0		0	0
C à C →				0
B à B →			0	

Les évolutions d'état en état de cette programmation ne se produisent que dans la phase (6) en synchronisme avec LcaT (lecture caractère Terminée). C'est à dire si $CR E_{E_3}$ ou $CR E_{2g_1}$ sont vrais.

b) En mise en place bande (phases (2) et (8))

Le tableau des états stables en fonctionnement dans ces phases, impliqués par les phases d'évolution de SP 8- SP 9. (Mise en place bande E_{8a_1} , lecture en avance ou en retour, bande prête exclues) est le suivant :

Etats stables possibles en phase (2) ou (8)	$P \rightarrow P$	$P_i \rightarrow P_i$	MAV	MRe
Mise en place bande pas à pas programmée	1	0	0	0
Avance bande en pas à pas	1	0	1	0
Retour bande en pas à pas	1	0	0	1
Mise en place bande pièce à pièce programmée	0	1	0	0
Avance bande en pièce à pièce	0	1	1	0
Retour bande en pièce à pièce	0	1	0	1
Etat stable hors des phases (2) ou (8)	0	0	0	0

le Tableau des interdits est représenté ci-contre :

P à P →				
Pi à Pi →				
MAV →				0
MRe →			0	

Les évolutions d'état en état de cette programmation ne se produisent que dans les phases globales (2) ou (8) en synchronisme avec LcaT (lecture caractère terminée). C'est à dire si $CR E_{8c_1}$, $CR E_{8d_1}$, $CR E_{9c_1}$, $CR E_{9d_1}$ sont vrais.

4°) L'unité de lecture de un caractère

C'est la machine logique qui effectue l'ensemble des phases d'évolution des ordinogrammes SP8 -SP0 (M01) et de la partie des phases d'évolutions de SP2(M01) comprises entre E_{2a_1} et E_{2e_3} ou E_{2g_1} incluses.

C'est une machine cyclique d'état initial et d'état final LcaT qui ne décrit le cycle LcaT LcaG que si l'on est en lecture en cours LC dans les phases globales (2) (6) (8). Dans tout autre état global elle est bloquée à l'état LcaT.

.....

5°) L'unité d'arrêt extérieur machine :

C'est la machine logique qui effectue l'ensemble des phases d'évolution de l'ordinogramme SP 12(M01) (planche 5-9)

L'ordinogramme SP 12(M01) ne comportant aucune bande incidente ou émergente, l'unité d'arrêt extérieur est une machine autonome que nous étudierons de suite.

Considérons donc l'ordinogramme SP 12(M01) planche 5-9.

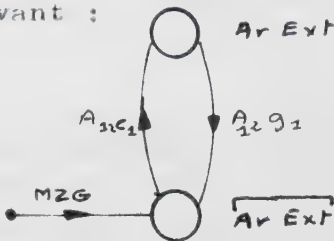
Nous avons deux phases d'évolution E_{12c_1} , E_{12g_1} ; l'unité ne peut se trouver que dans deux états qui sont :

arrêt extérieur Noté : Ar Ext
pas arrêt extérieur Noté : Ar Ext

Nous remarquerons que la mise sous tension de la machine peut être dans l'état Ar Ext ou Ar Ext (forcé par MZG) .

Le graphe des évolutions est le suivant :

A_{12c_1} est l'autorisation de l'évolution Arrêt extérieur.
 A_{12g_1} est l'autorisation de l'évolution Acquitement de l'arrêt extérieur.



Etudions le chemin A_{12c_1} , nous éliminerons la boucle $CR_{12c_1} A_{12c_1}$ qui est une boucle stable et non une boucle d'évolution.

La symbolisation est précisée sur l'ordinogramme.

M M01=1 signifiant que le mot Stop Non optionnel est en mémoire.

Nom de l'action :		MPB	USA	Ug	MM01	BP MPB	BP StopM	BP Ar Ug	BP AV	BP PE	BP ImpU
En mise en place bande	arrêt par Stop bande	1	1	1	1	1	1	1	1	1	1
	arrêt par Stop machine	1	1	1	1	1	1	1	1	1	1
	arrêt par Arrêt d'urgence	1	1	1	1	1	1	1	1	1	1
En usinage semi-automatique	arrêt par Stop machine	0	1	1	1	1	1	1	1	1	1
	arrêt par Arrêt d'urgence	0	1	1	1	1	1	1	1	1	1
En usinage automatique	arrêt par Stop machine	0	0	1	0	1	1	1	1	1	1
	arrêt par Arrêt d'urgence	1	1	1	1	1	1	1	1	1	1

$A_{12c_1} = BP \text{ Ar Ug} + BP \text{ Stop M. DA. } \overline{M \text{ M01}} + BP \text{ Stop B. MPB}$

• Etudions le chemin $A_{12}g_1$, il vient directement :
 $A_{12}g_1 = \overline{BP Ar Ug}$. (x); recherchons x.

Nous avons $x = 1$ pour tous les trajets précisés ci-contre, $\boxed{1-1}$ signifiant que l'on a : $\overline{BP Av + BP Re}$

MPB	AMOI	BPMPB	BPMPM	BP Av	BP Re	BP Imp U
1	1	0	0	1	1	1
1	0	0	1	1	1	1
1	1	0	0	1	1	1
1	0	0	1	1	1	1
0	1	1	0	1	1	1
0	1	1	1	1	1	1

D'où :

$$A_{12}g_1 = \overline{BPAr Ug} \cdot \left[\overline{MPB} \cdot \overline{BP StopB} \cdot (BP Av + BP Re) + \overline{MPB} \cdot BP Imp U \cdot \overline{MD MOI} \cdot \overline{BP Stop M} \right]$$

L'intersection des chemins $A_{12}g_1$ et $A_{12}c_1$ étant vide :
 $Ar Ext = \overline{MZG} \cdot \overline{A_{12}g_1} \cdot (A_{12}c_1 + Ar Ext)$

Remarques :

- l'on aurait pu concevoir un système différent, plus économique tel que $A_{12}g_1$ soit conservé et que :

$$A'_{12}g_1 = \overline{A_{12}a_1} \cdot [\overline{MPB} (BP Av + BP Re) + \overline{MPB} \cdot Imp U]$$

$$A'_{12}g_1 = \overline{A_{12}a_1} \cdot Y.$$

Réalisé avec une mémoire prioritaire à la mise à un, le circuit est moins coûteux, son équation est :

$$Ar Ext = \overline{MZG} \cdot [Ar Ext \cdot Y + A_{12}a_1]$$

- le signal $Ar Ext$ est un signal type alarme, nous le distribuerons dans la machine au plus près des fonctions logiques dont il doit interdire l'évolution, indépendamment de toute considération de minimisation.

6-3-2- Introduction à l'étude des unités :

Nous allons dans les deux chapitres suivants étudier les unités fonctionnelles et déterminer les équations logiques binaires des différentes unités ci-dessus (unité de synthèse globale exclue)

Nous ne développerons pas toutes les équations de tous les signaux comme on est conduit à le faire afin d'établir les plans; nous négligerons aussi certains détails afin de ne pas trop alourdir l'étude; nous nous sommes permis d'écrire les équations en fonction du symbole affecté à un état interne et non en fonction des signaux logiques, la mise en équation définitive étant classique (problème combinatoire).

Nous n'étudierons pas le problème de la conversion de ces équations en schémas logiques établis à l'aide d'opérateurs logiques imposés par un standard des opérateurs utilisables.

.../...

1°) Nous commencerons par étudier l'unité de lecture de un caractère, c'est à dire l'ensemble des évolutions centrales du graphe. Il n'existe aucune boucle à l'intérieur du graphe de cette unité (graphe en réseau orienté). De plus tous les chemins incidents ou émergents au réseau sont incidents ou émergents respectivement des sommets initiaux et du sommet final. Deux états essentiels sont associés à cette machine LcaC (Lecture caractère en cours) et lecture caractère terminée LcaT.

2°) Nous poursuivrons par l'étude de la programmation des états globaux puis par la programmation manuelle du mode de lecture.

7 - UNITE FONCTIONNELLE
de
LECTURE de UN CARACTERE

	pages :
7-1- <u>Etablissement du graphe aux phases d'évolution</u>	131
7-1-1- Phases d'évolution de même application sémantique	131
7-1-2- Graphe des phases d'évolution	132
7-2- <u>Les sous-unités classiques.</u>	133
7-3- <u>La syntaxe de lecture</u>	136
7-3-1- réduction du graphe des phases d'évolutions	136
7-3-2- Etude des chemins	138
7-3-3- Sous-unité de mise en mémoire de la syntaxe	142
7-4- <u>Unité logique de synthèse de lecture de un caractère.</u>	143
7-4-1- Généralités relatives à la machine étudiée	144
7-4-2- Schéma de principe des séquences des ordres	147
7-4-3- Elaboration des ordres et des compte-rendu d'évolutions	147
7-4-4- Etude du compteur de répartition.	150
7-5- <u>Conclusions.</u>	154

INTRODUCTION

Nous nous proposons de déterminer dans ce chapitre la machine logique dite unité fonctionnelle de lecture d'un caractère. Elle permettra de décrire l'ensemble des phases d'évolution des ordino-grammes.

- de rupture de programme en mise en place bande (SP8-SP9/M01 Planche 5-7)

- de lecture de un caractère en usinage automatique (SP2/M01 Planche 5-4) ; de la phase < lecture de un caractère en avance > aux phases < lecture caractère terminée > .

Nous commencerons par établir le graphe aux phases d'évolutions de l'unité de lecture de un caractère.

Nous présenterons rapidement, sans les étudier, les sous-unités logiques réalisant les différentes phases du graphe. Ces machines logiques étant classiques, après avoir précisé leur but, nous indiquerons leurs différents ordres de commande.

Les sous-unités ci-dessus sont connues de part leurs équations les ordres de commande et les séquences qu'ils doivent respecter. Il nous reste à déterminer la logique de synthèse permettant lorsque l'on émet les autorisations de lire un caractère en Avance ou en Retour de décrire les phases d'évolutions.

- Pour ce faire, nous réduirons le graphe, nous déterminerons les chemins afin que l'on n'aie pas d'évolutions simultanées, cela nous conduira à établir la sous-unité de syntaxe.

- Puis, tenant compte de tous les ordres nécessaires pour les sous-unités, nous établirons les séquences qu'ils doivent respecter dans chaque cas de syntaxe. Nous serons conduit à l'unité de synthèse de lecture de un caractère pilotée par un compteur, évoluant en synchronisme avec un signal d'Horloge (dit Horloge de l'unité d'entrée), à condition que les ordres corrects correspondant aux séquences soient émis et reçus par les sous-unités fonctionnelles étudiées ci-dessus.

7-1- Etablissement du graphe aux phases d'évolution de l'unité de lecture de un caractère.

Reprenons les ordinogrammes SP2 (M01), SP8 (M01), SP9 (M01) et plutôt que de représenter un ordinogramme réduit dans lequel nous aurons fusionné toutes les phases d'évolution identiques (même application sémantique), et étudié les enchainements entre ces phases, nous allons représenter un graphe des phases d'évolution que nous allons déterminer.

7-1-1- Recherchons les phases d'évolution ayant la même application sémantique dans les trois ordinogrammes SP2 - SP8 - SP9 (M01) entre la phase de lecture de un caractère (en Avance ou en Retour) et la phase lecture terminée ; remplaçons le nom de la phase par un numéro, afin d'alléger les écritures
Faisons de même pour les sous-programmes SP10 et SP11 qui sont inclus dans SP2 - SP8 - SP9 (M01).

Nous sommes conduits aux tableaux ci-dessous.

	Applications sémantiques	Indexation des phases dans :	
		SP 10	SP 11
1	Impulsion d'avance lecteur	a ₁	-
2	Impulsion retour lecteur	-	a ₁
3	Echantillonnage contacts	b ₁	b ₁
4	Mise en mémoire tampon	c ₁	c ₁
5	Fin de l'échantillonnage des contacts	d ₁	d ₁

	Applications sémantiques	Indexation des phases dans :		
		SP 2	SP 8	SP 9
6	Erreur de syntaxe	c ₁	b ₁	b ₁
7	Mise en mémoire des mots (SP3)	b ₁ ;b ₃ ;b ₆	-	-
8	Mise en mémoire des adresses cotées	c ₃	-	-
9	Mise en mémoire des adresses non cotées	c ₄	-	-
10	Mise en mémoire des signes	c ₅	-	-
11	Mise en mémoire des chiffres	c ₆	-	-
12	Lecture bloc terminé	c ₁₀	-	-
13	Rupture AV début programme	-	b ₆	-
14	Rupture Re début programme	-	-	b ₆
15	Mise en mémoire FB	c ₉ ;d ₃	c ₂	b ₄
16	Mise en mémoire BL	-	b ₄	c ₂
17	Mise à zéro FB et BL	-	b ₅	b ₅
18	Lecture caractère terminé	d ₁ ;e ₃	c ₁ ;c ₁	c ₁ ;d ₁

7-2- Les sous-unités classiques :

Avant d'étudier les chemins du graphe précédent nous allons déterminer les sous-unités logiques qui réalisent les phases d'évolutions. Nous connaissons alors toutes les machines de queue de l'unité de lecture d'un caractère, il sera alors possible d'effectuer la synthèse en fonction des ordres logiques pilotant ces sous-unités.

Ces machines sont simples et classiques, nous n'en développerons pas les équations, ni les schémas, nous préciserons succinctement leur but, leurs entrées et leurs sorties, nous récapitulerons leurs ordres de commande afin de bien connaître les sorties de l'unité de synthèse de un caractère.

Afin de suivre les interconnexions de ces machines le lecteur se reportera à la planche 7-1 jointe.

7-2-1- Les sous-unités de lecture de un caractère de la bande programme :

Ces sous-unités permettent de décrire les phases 1, 2, 3, 4, 5 du graphe.

1°) Le lecteur de bande (phase 1 et 2)

Les sorties sont les états de 8 contacts notés C_{ei} , tels que si la variable e_i de la bande programme est à UN logique (un trou dans la bande programme sur la piste i), nous avons $C_{ei} = 1$

Deux ordres permettent de commander les déplacements de la bande de un pas dans le sens Avance ou Retour.

[Ordre d'Avance : mp Av
Ordre de Retour : Imp Re

2°) L'échantillonnage de l'état des contacts (phases 3 et 5 :

Les rebondissements des contacts, les problèmes d'interfaces entre le lecteur et la logique, imposent d'échantillonner les états C_{ei} des contacts par un ordre d'échantillonnage : Echamt, tel que les sorties soient $e_i = C_{ei} \cdot \text{Echamt}$.

3°) Le registre tampon de mise en mémoire du caractère lu (phase 4) :

Le registre permet de mettre en mémoire le caractère échantillonné ci-dessus dans 8 bascules Me_i .

Ce registre permet aussi de décaler vers la gauche les chiffres lus c'est à dire les digits Me_4, Me_3, Me_2, Me_1 (voir registre des entiers).

.../...

Les ordres de commande sont :

[Mise à zéro du registre : M2T
 Mise à N du registre : MnT
 Impulsion de décalage : Décal (actif par le flanc de descente).

7-2-2- Les sous-unités de détection de syntaxe :

Elles permettent d'élaborer les signaux correspondant aux blocs de décisions des ordinogrammes SP2 (M01), SP8 (M01) et SP9 (M01) correspondant aux chemins émergents de la phase 5 du graphe. Dans l'ordre l'on doit vérifier que :

1°) La parité est correcte :

C'est une unité combinatoire à un signal de sortie :
 [Pcor telle que Pcor = 1 si le nombre de variables à l'état 1, en mémoire dans le registre tampon est pair (en code I.S.O).

2°) Le caractère lu (détection du caractère lu) appartient à l'alphabet $\mathcal{A}_{M01}$, c'est à dire qu'il ne peut être que :

<une adresse cotée >	: DAC
<une adresse non cotée >	: DANC
<un signe >	: DSgn
<un chiffre >	: Dch
<un blanc >	: DBL
<un Fin de bloc >	: DFB
<un caractère de mise en page >	DMP

3°) Le caractère lu au pas précédent (prédécesseur) est correct :

Le graphe des prédécesseurs adjacents du paragraphe 5-3-2-4 précise l'ensemble des prédécesseurs nécessaires. Chacun étant déterminé par le caractère lu au pas précédent il est nécessaire de les mettre en mémoire dans un registre à 7 états.

<précédé de adresse cotée >	MDAC
<précédé de adresse NON cotée >	MDANC
<précédé de signe >	MDSgn
<précédé de chiffre >	MDCh
<précédé de Blanc >	MDBL
<précédé de Fin de Bloc >	MDFB
<pas de prédécesseur >	Ø

Les ordres de commande du registre sont :

[Mise à zéro du registre : MZP
 Mise en mémoire du type de prédécesseur : τ_{nP}

.....

7-2-3)- Les sous-unités de mise en mémoire des caractères :

Un mot est formé d'un ensemble de caractères lus les uns après les autres à chaque pas. Il est nécessaire d'effectuer la mise en mémoire de chaque caractère au fur et à mesure de la lecture, dans des mémoires auxiliaires, tant que la lecture du mot n'est pas terminée, c'est à dire tant que l'on n'a pas mis le mot lu en mémoire dans la mémoire des mots. Au fur et à mesure de la lecture d'un mot l'on effectue :

1°) La mise en mémoire de l'adresse dans le registre dit mémoire des adresses (Phases 8 et 9)

C'est un registre à 12 états notés : MDN ; MDT ; MDS. MDF ; MDX ; MDY ; MDI ; MDJ ; MDG ; MDM et un état initial dit « pas d'adresse » noté $\emptyset$.

Les ordres de mise en mémoire sont :

[Mz(A S E) : mise à zéro des adresses, signes et chiffres
MnAd : mise à n des adresses.

2°) La mise en mémoire du signe dans le registre dit mémoire des signes (Phase 10)

C'est un registre à 2 états notés : MD+ ; MD- ; et un état pas de signe noté $\emptyset$.

Les ordres de mise en mémoire des signes sont :

[Mz (A, S, E) : mise à zéro des signes
Mn Sgn : mise à n des signes.

3°) La mise en mémoire des Chiffres (phase 11)

Dans un mot on lit un entier, c'est à dire une succession de chiffres (au maximum 5, pour la machine M01). Les mots étant d'un format variable le poids du premier chiffre lu est inconnu, le poids du dernier est connu, c'est le chiffre des unités.

L'on est conduit à décaler chaque chiffre lu (en mémoire tampon) vers la gauche dans le registre des entiers.

L'ensemble formé par les 5 décades du registre des entiers et les 4 bascules (Me_4 , Me_3 , Me_2 , Me_1) du registre tampon est un registre à décalage à entrées en parallèle et à sorties en parallèle, piloté par :

[un ordre de décalage : Décal
un ordre de mise à zéro : Mz (A, S E)

Mettre en mémoire des entiers un chiffre lu, revient à décaler 4 fois vers la gauche les digits (Me_4 , Me_3 , Me_2 , Me_1) en émettant 4 fois Décal.

.....

7-2-4- Les sous-unités de mise en mémoire des mots : phase 7

Les études syntaxiques du chapitre 3 développées lors de l'étude des ordinogrammes font apparaître dans l'ensemble des mots les sous-ensembles suivants :

- 1°) Les mots dont l'application sémantique est la même, quelle que soit la valeur de l'entier.

Rappelons ces mots, leurs adresses et leur format.

Familles des mots	adresse	format
Numéro de Bloc	N	3
Fonction d'outil	T	2
Vitesse de broche	S	4
Vitesse d'avance	F	4
Mouvement de translation	X Y Z	5
Paramètres auxiliaires de contour- nage	I J	5

Ce sont des registres classiques, actifs pour X,Y,Z,I,J et passifs pour les autres adresses. Ils comportent autant de décade Décimale codé binaire que le format des mots l'exige.

Ils sont initialisés par les ordres suivants :

- [Mise à zéro des mots MzM
- [Mise à n des mots MnM

Ces ordres sont communs à tous les registres. Chaque registre est sélectionné par l'adresse en mémoire des adresses.

- 2°) Les mots dont l'application sémantique est fonction de la valeur de l'entier.

Ce sont les mots ci-après :

Familles des mots	mots	mot initial
Trajectoire à usiner	G01 G02 G03 G04	pas de mot
cycles fixes	G80 G81 G82 G83 G84	G80
serrage pièce	M68 M69	M69
serrage table	M78 M79	M79
arrosage	M08 M09	M09
rotation broche	M03 M05 M04	M05
arrêt machine	M01 M03	M01
rupture programme	M00 M02 M10 M30 M12	M12

Chaque partie ci-dessus peut comporter au plus un mot en mémoire. A une partie nous associerons un registre passif.

Ils sont initialisés par les mêmes ordres que les registres précédents, sélectionnés par l'adresse en mémoire et l'ensemble des entiers leur appartenant.

Nota : Les mots M00|M02|M10|M12 impliquant des ruptures de séquence dans l'ordinogramme SP1 (M01), l'on effectue la mise à l'état initial M12 par des ordres extérieurs (MzM00) émis par l'unité de programmation.

7-3- La syntaxe de lecture :

Les propriétés des sous-unités ci-dessus nous permettent de réduire le graphe, d'étudier les équations booléennes des chemins possibles, de rechercher les états prédécesseurs rendant ce graphe déterministe. Nous en déduirons la sous-unité de mise en mémoire de la syntaxe.

7-3-1- Réduction du graphe des phases d'évolution :

- 1°) Les phases d'évolutions 8 et 9 du graphe sont iden-

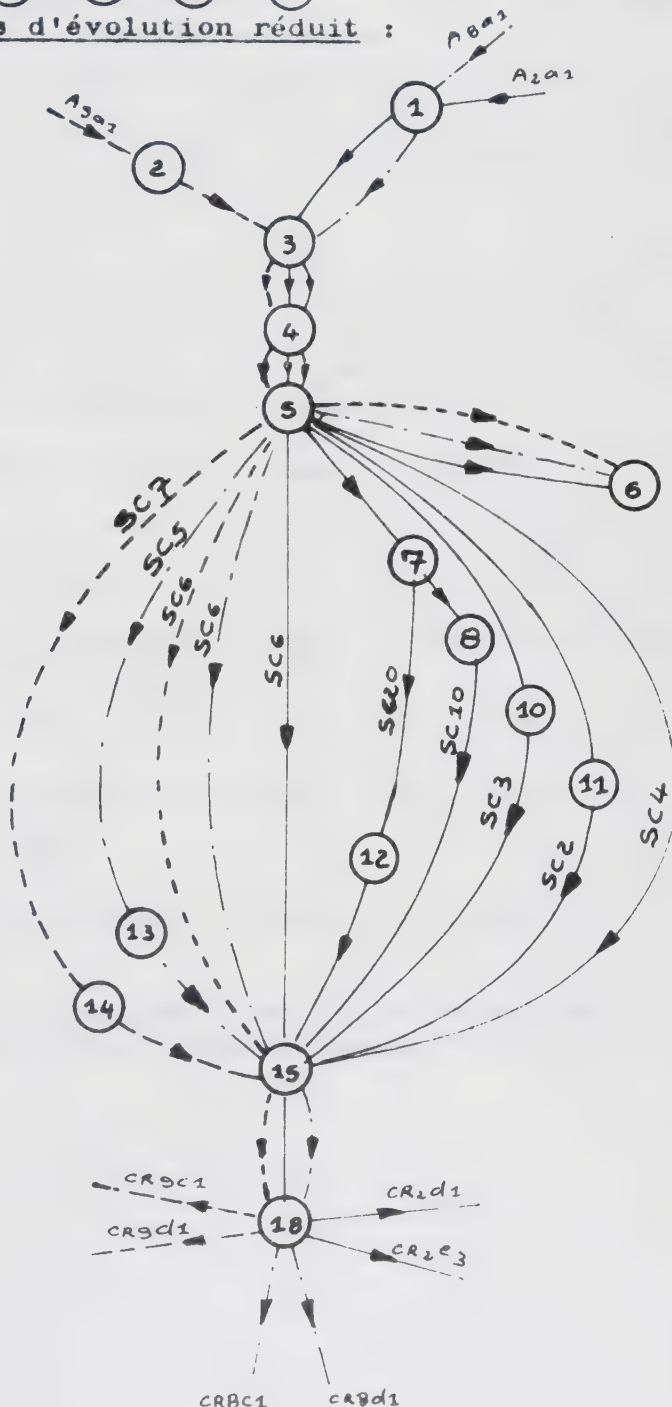
tiques (un seul registre des adresses), l'on peut fusionner ⑧ et ⑨ en ⑧. Reste le problème du chemin, compatible avec l'ordinogramme SP2. Nous le ramènerons à un chemin ⑤ → ⑦ après avoir remarqué que l'on peut effectuer la mise en mémoire du mot lu précédemment (dans notre cas aucun mot) les registres de mémoire adresse, signe, chiffres étant à l'état < pas d'adresse >

2°) Il est possible d'effectuer la mise en mémoire de FB, BL et la mise à zéro à l'aide du registre des prédécesseurs. Dans ce cas ⑮, ⑯, ⑰ seront réduits en ⑮ appelé phase d'évolution de mise en mémoire des prédécesseurs. Il nous reste la mise en mémoire de MDAC, MDANC, MDSgn, MDCh, nous introduirons la phase ⑮ sur les chemins

⑧ → ⑮ ; ⑨ → ⑮ ; ⑩ → ⑮ ; ⑪ → ⑮

3°) Grappe aux phases d'évolution réduite :

- ① Imp Avance
- ② Imp Retour
- ③ Echantillonnage
- ④ Mise en mémoire tampon
- ⑤ Fin échantillonnage
- ⑥ Erreur de syntaxe
- ⑦ mise en mémoire Mot + Mz (A,S,E)
- ⑧-⑨ Mise en mémoire adresse
- ⑩ Mise en mémoire signe
- ⑪ Mise en mémoire chiffre
- ⑫ Lecture bloc Terminée
- ⑬ Rup. Avance
- ⑭ Rup. Retour
- ⑮ Prédécesseurs
- ⑯ Lecture terminée



7-3-2- Etude des chemins du graphe réduit aux phases d'évolutions

1°) Etablissement d'un graphe déterministe :

Nous ne devons pas avoir d'évolutions simultanées dans ce graphe. Nous dirons comme pour les graphes des états internes des machines de MOORE ou de MEALY que ce graphe doit être déterministe. (Propriétés des chemins émergents d'un même sommet).

Nous allons étudier les équations booléennes des chemins qui ne sont rien d'autre que les équations d'autorisations d'évolutions de l'ordinogramme que l'on pourrait construire à partir de ce graphe à l'aide du tableau (page 139)

Pour toutes les évolutions simultanées nous rechercherons des < états > prédécesseurs dans l'ensemble des ordinogrammes qui permettent de rendre le graphe déterministe.

L'ensemble des chemins correspondant aux ordinogrammes SP8-SP9 est rendu déterministe en ajoutant les états prédécesseurs MAV (Mémoire Avance) et MRE (Mémoire retour) impliqués par les phases d'évolutions Ed_1 ou Ed_3 , et Ed_2 ou Ed_4 des ordinogrammes SP8 et SP9 (M01).

L'ensemble des chemins correspondant aux ordinogrammes SP2 - SP8 - SP9 sera rendu déterministe en ajoutant les états prédécesseurs MPB (mise en place bande) pour SP9 et SP8 et UA. MPB pour SP2 (MAV et MRE étant indifférents bien que le lecteur ne puisse être qu'en Avance bande).

Nota : Nous remarquerons que dans le tableau :

- ① Parmi les états de la détection du type de caractère lu, un seul état peut être à un, les autres à zéro ; nous n'indiquerons pas tous les zéros.
- ② Même remarque que ci-dessus, quant à l'état de la mémoire du type de caractère prédécesseur.
- ③ $\epsilon_{M01} = DAC + DANC + DSgn + DCh + DBL + DFB$
- ④ La détection de l'adresse N signal DN ne peut être à un que si $DANC = 1$
 $\text{si } \overline{DANC} : \longrightarrow \overline{DN}$
- ⑤ Lorsque l'état d'un signal n'a pas d'influence nous le noterons par : -
- ⑥ Nous remarquerons finalement que le chemin ⑤ → ⑥ n'est vrai que si aucun autre chemin émergent de ⑤ n'est vrai.

.../...

TABLEAU DE LA SYNTAXE

Syntaxe	Ordinogrammes	chemins du graphe réduit		chemins du graphe initial					Détection du Type de caract. -ère Lu.					Type caractère prédécesseur					« Etats » prédéces- seurs.							
		de	di	de	di	Pcor	Edu _{moz}	DN	DMP	DAC	DANC	Dsgn	Dch	DBL	DFB	MDAC	MDANC	MDsgn	MDch	MDBL	MDFB	UA	MPB	MAV	MRE	
SC10	SP2	5	7	5	7	1	1			1									1			1	0	-	-	
SC10		5	7	5	7	1	1	0			1								1			1	0	-	-	
SC20		5	7	5	7	1	1							1					1		0	1	0	-	-	
SC10		5	7	5	7	1	1	1			1										1	1	0	-	-	
SC3		5	10	5	10	1	1					1				1							1	0	-	-
SC2		5	11	5	11	1	1						1					1					1	0	-	-
SC2		5	11										1				1						1	0	-	-
SC2		5	15										1						1				1	0	-	-
SC4		5	18	5	18	1	0		1								-	-	-	-	-	-	1	0	-	-
SC5		5	15	5	15	1	1								1						1		1	0	-	-
SC6	SP8	5	15	5	16	1	-							1	0	-	-	-	-	-	-	-	1	1	0	
SC6		5	15	5	17	1	-		-	-	-	-	-	0	0	-	-	-	-	-	-	-	1	2	0	
SC5		5	13	5	13	1	-								1					1		-	1	1	0	
SC6	SP9	5	15	5	15	1	-								1	-	-	-	-	-	-	-	1	0	1	
SC6		5	15	5	17	1	-		-	-	-	-	-	0	0	-	-	-	-	-	-	-	1	2	0	
SC7		5	14	5	14	1	-							1							1	-	1	1	0	
SC10	SP2	7	8	7	8	1	1			1									1			1	0	-	-	
SC10		7	8	7	9	1	1	0			1								1			1	0	-	-	
SC20		7	12	7	12	1	1							1					1		0	1	0	-	-	
SC10		7	8	5	9	1	1	1			1										1	1	0	-	-	

2°) Equations de syntaxe d'évolution :

Partant de la phase (5), nous allons rechercher les équations des chemins émergents de (5) qui impliquent un trajet déterminé dans le graphe.

Les trajets possibles de parcours du graphe sont les suivants :

a) Trajet (5) → (7) → (8) → (15) → (18)

l'équation du chemin (5) → (7) impliquant le trajet ci-dessus est :

$$SC_{10} = Pcor. UA. \overline{MPB} \left[DAC. MDCh + DANC. \overline{DN}. MDCh + DANC. DN. MDFB \right]$$

cette équation est valable de (5) à (15)

b) Trajet (5) → (7) → (12) → (15) → (18)

l'équation du chemin (5) → (7) impliquant le trajet ci-dessus est :

$$SC_{20} = Pcor. UA. \overline{MPB}. DFB. MDCh$$

cette équation est valable de (5) à (15)

c) Trajet (5) → (11) → (15)

$$SC_3 = Pcor. UA. \overline{MPB}. DCh (MDAC + MDANC + MDCh).$$

Elle est valable de (5) à (11). Pendant la phase (11) l'on décale les chiffres (en particulier les informations Me_4, Me_3, Me_2, Me_1 du registre Tampon) La parité devient incorrecte (par exemple : chiffre 8 ou 9 décalés une fois vers la gauche.).

La détection de chiffre DCh devient incorrect (exemple: 6 et 7 décalé 1 fois à gauche) et, est correcte pour tous les chiffres après 3 décalages à gauche.

d) Trajet (5) → (18)

$$SC_4 = Pcor. UA. \overline{MPB}. DMP$$

e) Trajet (5) → (13) → (15) → (18)

$$SC_5 = Pcor. MPB. MAV. DFB. MDBL$$

valable de (5) à (15)

f) Trajet (5) → (15) → (18)

$$SC_6 = Pcor. \left[\begin{array}{l} UA. \overline{MPB}. DFB. MDFB \\ + MPB. MAV. \overline{DFB}. MDBL \\ + MPB. MRC. \overline{DBL}. MDFB \end{array} \right]$$

valable de (5) à (15)

.../...

g) Trajet (5) → (14) → (15) → (18)

$SC_7 = Pcor . MPB . MR . DBL . MDFB$

valable de (5) à (15)

h) Trajet (5) → (6)

Il est décrit lorsque l'on n'a aucune équation ci-dessus vraie. L'équation de l'erreur syntaxe est :

$$E_{sc} = SC_{10} + SC_{20} + SC_2 + SC_3 + SC_4 + SC_5 + SC_6 + SC_7$$

3°) Equation des compte-rendus d'évolution en phase d'évolution (18) terminée :

Nous avons :

$$CR_2 d_1 = CR (18) . \overline{LBT}$$

$$CR_2 e_3 = CR (18) . LBT$$

$$CR_8 c_1 = CR (18) . \overline{Rup Av}$$

$$CR_8 d_1 = CR (18) . Rup Av$$

$$CR_9 c_1 = CR (18) . \overline{Rup Re}$$

$$CR_9 d_1 = CR (18) . Rup Re$$

$LBT \leftrightarrow OUI$ < Lecture bloc terminée >

$Rup Av \leftrightarrow OUI$ < Rupture Avance début programme >

$Rup Re \leftrightarrow OUI$ < Rupture Retour début programme >

Ces prédécesseurs sont évidents (l'on se reportera aux ordinogrammes SP2 ; SP8 ; SP9 M01.)

4°) Conclusions :

Reprenons le problème de la détection de syntaxe des chiffres SC_3 et des ruptures de lecture (LBT , Rup Av, Rup Re) et résumons les caractéristiques de ces quatre «états» .

SC_3 a des aléas pendant la phase (11)

$SC_{20} \rightarrow (12) \rightarrow LBT$

$SC_5 \rightarrow (13) \rightarrow Rup Av$

$SC_7 \rightarrow (14) \rightarrow Rup Re$

Les ruptures de lecture sont en plus des états prédécesseurs des phases d'évolutions impliquées par lecture caractère terminée.

... ..

Nous choisirons d'effectuer la mise en mémoire de tous les cas de syntaxe en fin de la phase ⑤ (bien que le choix ne soit absolument pas minimal), dans un registre dit de syntaxe, d'états :



MSC20 ; MSC5 ; MSC7 seront utilisés comme états de ruptures de lecture. Dans ce cas les phases ⑫ , ⑬ , ⑭ sont ramenées en fin de la phase ⑤ .

7-3-3- Sous-unité de mise en mémoire de la syntaxe :

Nous avons convenu de mettre en mémoire les états suivants MSC10, MSC20, MSC2, MSC3, MSC4, MSC5, MSC6, MSC7 <pas de syntaxe> .

<pas de syntaxe> correspondant à l'erreur de syntaxe ⑥ .

• Les états MSC10 MSC5 MSC7 sont des états de rupture de séquences (après la phase d'évolution ⑮). Etant dans ces états nous reviendrons à l'état < pas de syntaxe > après la phase lecture caractère terminée sur ordres extérieurs (à l'unité de lecture de 1 caractère) :

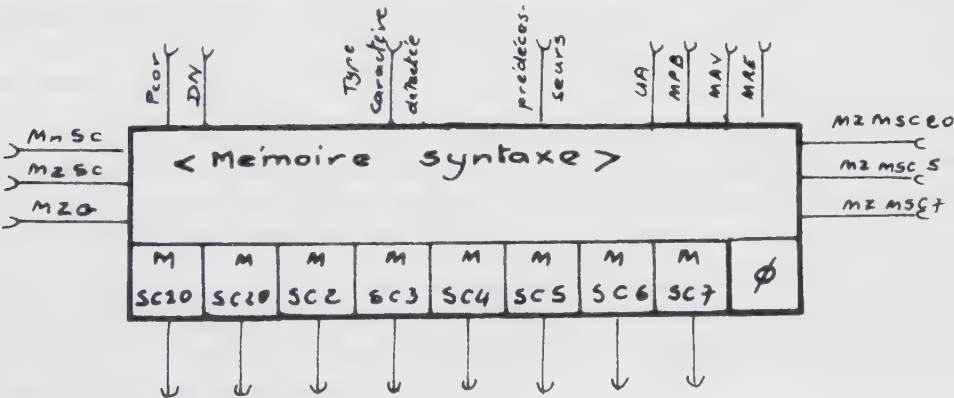
MZ MSC20 , MZ MSC5 , MZ MSC7

• La mise en mémoire des états de syntaxe s'effectuera par un ordre MnSc.

• La mise à l'état < pas de syntaxe > par l'ordre MZG classique.

• La mise à l'état < pas de syntaxe > (états MSC20, MSC5 MSC7 exclus) par un ordre MzSc.

Nous ne détaillerons pas cette sous-unité donnons simplement son schéma, les séquences étant classiques.



.../...

7-4- Unité logique de synthèse de lecture de un caractère :

Nous venons d'établir dans les paragraphes précédents :

- Tous les ordres de commande des différentes sous-unités et leurs séquences par sous-unité (correspondant à une phase d'évolution).
- L'enchaînement des phases d'évolutions dans le temps en fonction de tous les cas de syntaxe. (Rappelons que (12), (13), (14) ont été ramenées en fin de la phase d'évolution (5) par l'introduction de la mise en mémoire de la syntaxe).

Les autorisations d'évolutions sont :

$A_2 a_1$ pour la phase (1), d'états prédécesseurs connus :
UA. MPB. MAV

$A_8 a_1$ pour la phase (1), d'états prédécesseurs connus :
MPB . MAV

$A_9 a_1$ pour la phase (2), d'états prédécesseurs connus :
MPB . MRE

Nous allons déterminer maintenant l'unité de synthèse de lecture de un caractère, c'est à dire la machine logique qui émet tous les ordres (en respectant les séquences) nécessaires à l'enchaînement des phases d'évolutions dans tous les cas de syntaxe.

Plutôt que d'effectuer la synthèse formelle de cette unité par les méthodes classiques (séquences des entrées, séquence des sorties) nous avons utilisé un compteur R de répartition des ordres, synchrone par rapport à un signal d'horloge C_1 n'évoluant d'état en état que si les ordres corrects ont été émis, pour un état de la syntaxe donné .

Les séquences de tous les ordres, en fonction des états du compteur et de la syntaxe, ont été déterminées par l'établissement d'un schéma des séquences.

Nous noterons :

1°) Que l'unité logique de synthèse ne peut être qu'une machine du type synchrone (signal d'horloge C_1 de lecture). En effet ce choix est imposé par les caractéristiques des sous-unités que nous venons de déterminer.

- les registres de mise en mémoire sont à évolutions non consécutives (les sorties de chaque registre décrivent des états aléatoires pendant les phases d'évolutions du registre considéré).

- les circuits combinatoires ont les aléas classiques dus aux durées de propagation dans les chaines d'opérateurs, avec en plus les aléas dus aux transitions des registres.
- tous les ordres doivent avoir une durée fixe (machine à évolution non consécutive à niveau).

2°) Les ordres ne doivent pas avoir d'aléas

7-4 -1- Généralités relatives à la machine étudiée :

1°) Tableau des entrées et des sorties :

Signaux ou états des entrées				
Autorisations d'évolution {A}	Prédécesseurs {P}	Mémoire syntaxe {S}	Horloge {C}	Mise aux {C.I.}
A ₂ a ₁ A ₈ a ₁ A ₉ a ₁	UA MPB MAV MRÆ	MSC10 MSC4 MSC20 MSC5 MSC2 MSC6 MSC3 MSC7 pas syntaxe	C ₁	MZG

Signaux de sortie				
Compte-rendus d'évolution {CRE}	Ordres {O}			
CR ₂ d ₁ CR ₂ e ₃ CR ₈ c ₁ CR ₈ d ₁ CR ₉ c ₁ CR ₉ d ₁	Imp Av Imp Re Echant MzT MnT	MzSc MnSc MzM Mz(A,S,E) Mn Ad	MnSgn Décal MzP MnP	

2°) Principe de la machine :

Notons : a_n, p_n, s_i, c_j, o_m un état déterminé à une époque donnée des différents groupes d'entrées de la machine. Les états du compteur répartiteur seront notés r_i.

.../...

a) L'on pourrait concevoir une machine telle que le répartiteur d'ordre soit en "boucle ouverte", d'équations:

- en fonctionnement :

$$r_i, (c_j \cdot p_k \cdot s_1) \longrightarrow o_m$$

$$r_i, (\underline{c_1} \cdot a_n) \longrightarrow r_j$$

- à la mise aux conditions initiales par MZG

MZG $\longrightarrow r_0$: état initial du compteur répartiteur

MZG $\longrightarrow 0_{CI}$: état des ordres impliquant la mise aux CI des registres non forcés par MZG.

C'est la plus mauvaise machine que l'on puisse concevoir. Le compteur répartiteur évoluera quelque soient les ordres émis, quelque soient la syntaxe (s_k) et les prédécesseurs (p_k).

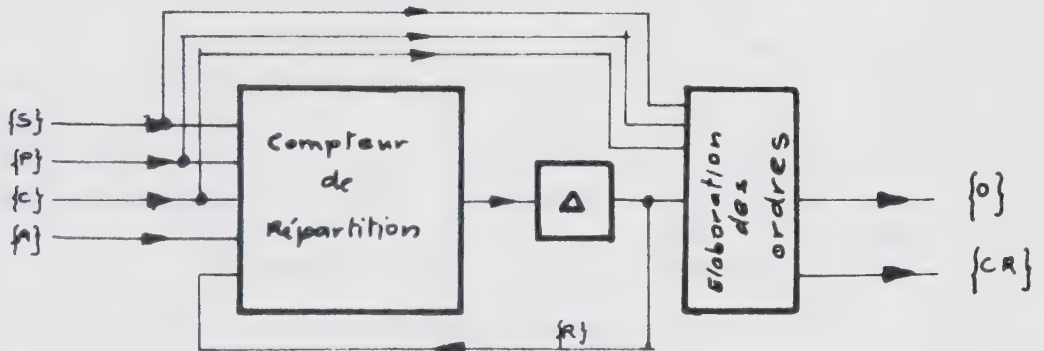
b) Choisissons donc une machine logique du type de Mealy :

- les équations de cette machine sont classiques. écrivons les pour les ordres et le compteur de répartition

$$r_i, (c_j \cdot p_k \cdot s_1) \longrightarrow 0_m \quad t \neq t_j \text{ de } \underline{c_1}$$

$$r_i, (\underline{c_1} \cdot a_n \cdot p_k \cdot s_1) \longrightarrow r_j \quad \text{époque } t_j \text{ de } \underline{c_1}$$

Cette machine se représente comme suit :



Nous avons cette fois une machine qui, si la mise en équation est correcte, n'évoluera pas si la syntaxe ou les prédécesseurs ne sont pas corrects, mais continuera d'évoluer quel que soit les ordres émis, reçus ou non reçus par les sous-unités.

...

- propriétés caractéristiques de cette machine :

① Les ordres ne doivent pas avoir d'aléas, les évolutions d'états de l'ensemble des ordres ne peuvent être que consécutives.

② Le compteur de répartition R ne peut être qu'à évolutions consécutives (nous le choisirons en code binaire réfléchi). En effet, si nous écrivons que les ordres ont un état constant o_m , sur un intervalle de temps T donné, la fonction d'ordre s'écrit :

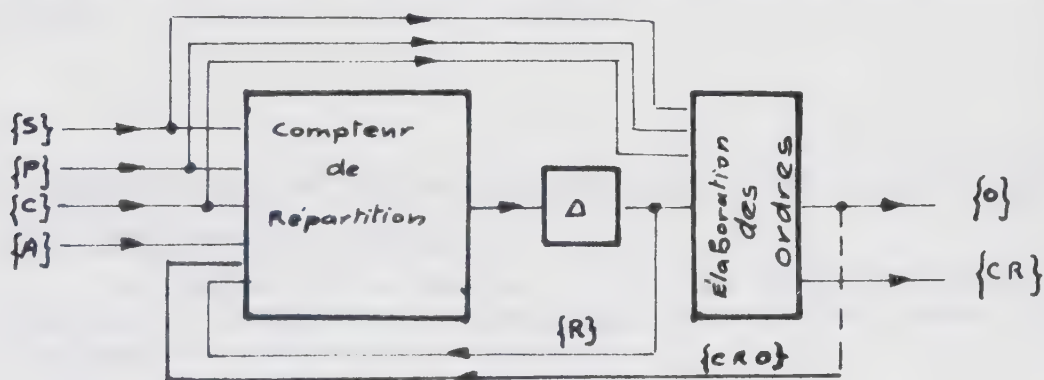
$$R_1, (C_j, p_K, s_1) \longrightarrow o_m$$

Si sur cet intervalle de temps R_1 est la séquence d'états : r_1, r_2, r_3 du compteur, ces états doivent être consécutifs ($r_1 = r_2 + r_3$), et c_j n'intervient pas.

③ Les états de syntaxe ne peuvent intervenir dans les équations des ordres que hors de la phase de mise en mémoire de la syntaxe (Δt après MSC).

- modifions notre machine afin qu'elle n'évolue plus si les ordres n'ont pas été émis. Cela se fera en introduisant les états des ordres dans la fonction d'évolution du compteur de répartition. Nous n'introduirons pas tous les états des ordres qui changent d'état entre deux instants consécutifs de déclenchement de C_1 .

Cette nouvelle machine se représente comme suit :



Ce n'est plus une machine minimale, mais nous contrôlons partiellement les ordres émis (échantillonnage aux époques t_i de C_1 , des signaux ayant changé d'état pendant l'état r_i du registre, précédent C_1).

.../...

Plutôt que d'étudier cette machine avec les signaux d'ordres {0}, nous utiliserons les signaux de compte-rendus d'ordre {CRO}. Ceci nous permettra d'augmenter la sécurité de fonctionnement de la machine (dans le sens de l'arrêt sur panne). En effet dans le cas des sous-unités logiques reliées à l'unité de lecture par l'intermédiaire d'un grand nombre de broches de connections (fiabilité faible), il est intéressant d'aller "chercher" le compte-rendu d'ordre dans la sous-unité considérée, plutôt qu'en bouclage direct sur l'unité de lecture.

Ce choix nous laissera toute liberté quant à la détermination du compte-rendu.

7-4-2- Schéma de principe des séquences des ordres :

Sur un schéma des séquences classique, faisant figurer les temps de retard entre la variation d'état d'un signal et la variation d'état qu'elle implique sur 1 seul autre signal, nous avons placé les différentes séquences des ordres, par phases d'évolution en fonction des entrées et des états du registre R.

Les séquences des ordres nous ont conduit à choisir un compteur répartiteur à 4 bascules R_1, R_2, R_3, R_4 .

Sur le schéma de séquence (page 148), nous n'avons pas représenté les prédécesseurs ; nous n'avons indiqué simplement qu'une autorisation d'évolution. Nous n'avons pas fait figurer la phase d'évolution du retour lecteur (Imp Re) afin de ne pas surcharger le schéma.

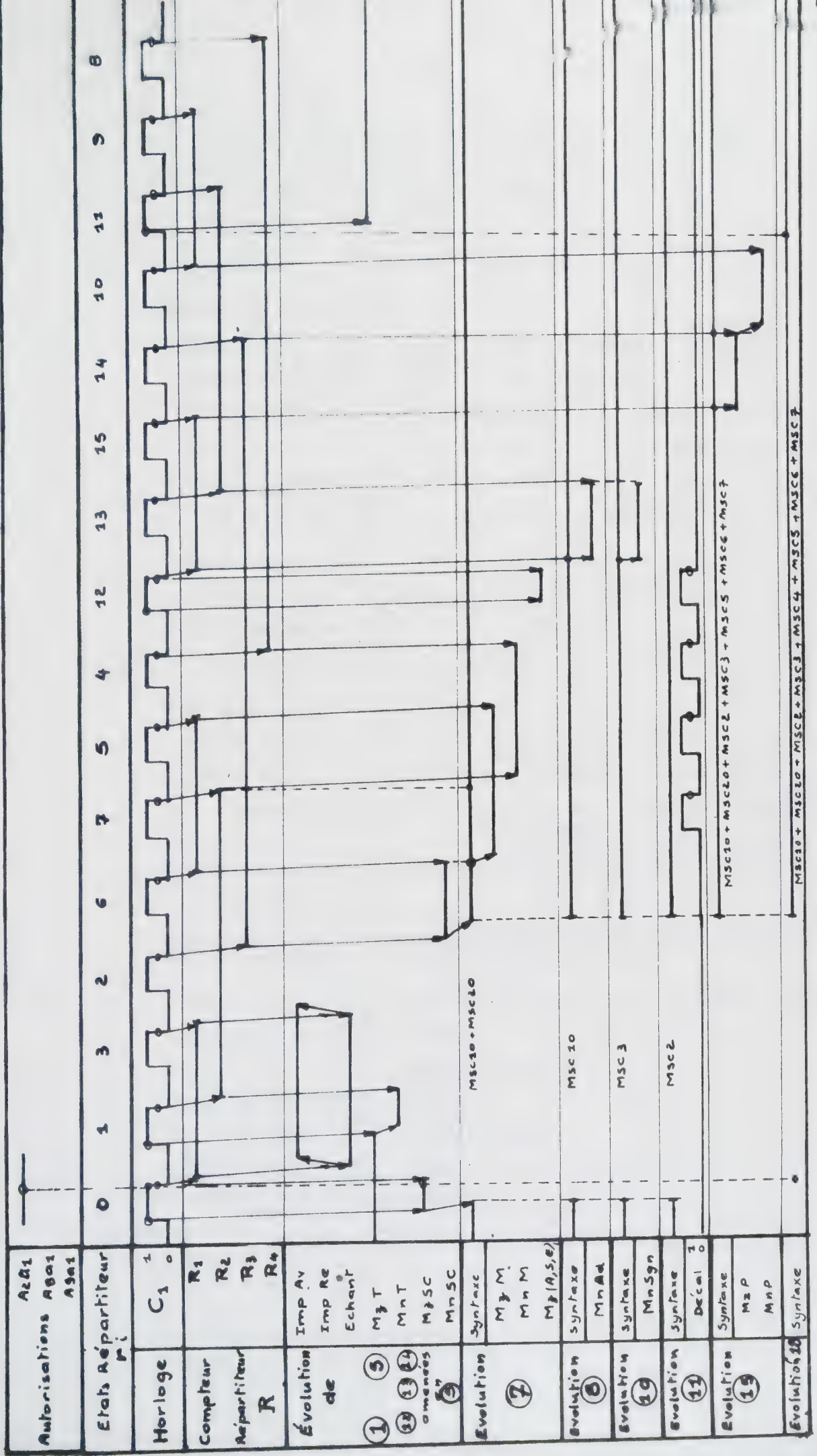
7-4-3- Elaboration des ordres et des compte-rendus d'évolution

1°) Equations des ordres :

Le schéma des séquences permet d'établir les équations des ordres. Effectuons la mise en équation par phases d'évolutions:

$$\begin{aligned}
 &\bullet \text{ Pour les phases de } \textcircled{1} \text{ à } \textcircled{5} \\
 &\text{Imp Av} = \text{CR échant.} \quad (\text{MAV.MBP} + \text{UA}.\overline{\text{MPB}}) \\
 &\text{Imp Re} = \text{CR échant.} \quad (\text{MRe} . \text{MPB}) \\
 &\text{Echant} = R_1 . \overline{R_3} . \overline{R_4} \\
 &\text{MzT} = \overline{B_{13}} \quad \dots/\dots
 \end{aligned}$$

TABEAU DES SEQUENCES



$$\text{avec } \begin{cases} sB_{13} = C_1 \cdot [CR \text{ Imp Av} + CR \text{ Imp Re}] \\ rB_{13} = C_1 \cdot R_1 \cdot R_2 \cdot R_3 \cdot R_4 + MZG \end{cases}$$

$$MnT = R_2 \cdot CR \text{ Echant} \cdot \overline{CRMzT}$$

$$MzSC = C_1 \cdot R_1 \cdot R_2 \cdot R_3 \cdot R_4 +$$

$$MnSC = R_1 \cdot R_2 \cdot R_3 \cdot R_4$$

● Pour la phase (7)

$$MzM = R_1 \cdot R_3 \cdot R_4 \cdot (MSC10 + MSC20) \cdot \overline{CR MnSC}$$

$$MnM = R_2 \cdot R_3 \cdot R_4 \cdot (MSC10 + MSC20)$$

$$Mz(A, S, E) = C_1 \cdot R_1 \cdot R_2 \cdot R_3 \cdot R_4 \cdot (MSC20 + MSC10) \cdot \overline{CR MnM} + MZG$$

● Pour la phase (8)

$$Mn Ad = R_1 \cdot R_2 \cdot R_3 \cdot R_4 \cdot MSC10 \cdot \overline{CRMz(A, S, E)}$$

Pour la phase (9)

$$MnSgn = R_1 \cdot R_2 \cdot R_3 \cdot R_4 \cdot MSC3 \cdot \overline{CRMz(A, S, E)}$$

● Pour la phase (11)

$$\underline{\text{Decal}_1} = R_2 \cdot R_3 \cdot C_1 \cdot MSC2$$

● Pour la phase (15)

$$MzP = R_1 \cdot R_2 \cdot R_3 \cdot R_4 \cdot A_1$$

$$MnP = R_1 \cdot R_2 \cdot R_3 \cdot R_4 \cdot A_1 \cdot \overline{CRMzP}$$

$$\text{avec } A_1 = MSC10 + MSC20 + MSC2 + MSC3 + MSC5 + MSC6 + MSC7$$

● Pour la phase (18) (voir l'équation de MzT phases (1) à (5)).

2°) Les équations des compte-rendus de la phase de lecture de un caractère terminée, s'écrivent :

$$CR_2 d_1 = 0 \cdot C_1 \cdot (MSC10 + MSC2 + MSC3 + MSC4 + MSC6)$$

$$CR_2 e_3 = 0 \cdot C_1 \cdot MSC20$$

$$CR_8 c_1 = 0 \cdot C_1 \cdot (MSC6 + MSC4)$$

$$CR_8 d_1 = 0 \cdot C_1 \cdot MSC5$$

$$CR_9 c_1 = 0 \cdot C_1 \cdot (MSC6 + MSC4)$$

$$CR_9 d_1 = 0 \cdot C_1 \cdot MSC7$$

$$\text{avec } 0 = R_1 \cdot R_2 \cdot R_3 \cdot R_4$$

7-4-4- Etude du compteur de répartition R :

Notre compteur a 16 états, indexons les exceptionnellement par la valeur décimale associée à l'état (r_0 à r_{15}), que nous repèrerons dans les équations par l'indice.

Nous allons dans l'ordre :

- Etablir une partition des états des chemins d'évolutions à l'aide du tableau des états des chemins.
- Etablir le tableau des évolutions aux partitions.
- Etablir les équations aux états du compteur et aux partitions des chemins.
- Etablir les équations du compteur.

1°) Partition des états des chemins d'évolutions :

L'on se reportera au tableaux des états des chemins page 151 .

Plutôt

que d'étudier les chemins , nous rechercherons une partition de ces chemins notée $\{Ep\}$, chaque partie étant repérée par P_i (de P_1 à P_{15}). Il restera les états tels que tous les signaux CR_0 soient à l'état zéro, les partitions seront établies pour les états des MSC_i (soulignés).

Nota : Nous n'avons pas tout noté sur ce tableau :

- Nous n'indiquerons que les 1 sur ce tableau, les 0 ne seront précisés que si lors du chemin précédent le signal était à l'état 1.

- L'état en mémoire de syntaxe est un seul MSC_i à l'état 1. Dans le cas de fonction (OU) des états, nous encadrerons les états 1 appartenant au {OU}.

- Valeur octaire inverse pour ces états OU :

$$T = \text{les trois signaux sont tels que: } T = s_1 + s_2 + s_3$$

A et C seront identifiés facilement.

2°) Tableau des évolutions aux partitions :

Nous avons déterminé les partitions dans le cas où tous les ordres sont à zéro.

$$P_{20} \text{ si } MSC_4 ; P_{21} \text{ si } (MSC_5 + MSC_6 + MSC_7)$$

$$P_{22} \text{ si } MSC_3 ; P_{23} \text{ si } MSC_2 ; P_{24} \text{ si } MSC_{20} ; P_{25} \text{ si } MSC_{10}$$

Nous n'indiquerons sur ce tableau que les changements d'états (dans une case, rien d'écrit signifie stable).

.../...

Parties	Valeur octaire Inverse							Chemins		Poids de la valeur octaire																									
										1°		1°		1°		10 ³		10 ⁴		10 ⁵		10 ⁶		10 ⁷		10 ⁸									
										Ordres émis Notes: CR																		Etats Mémoire syntaxe							
{E _P }	10 ⁸	10 ⁷	10 ⁶	10 ⁵	10 ⁴	10 ³	10 ²	10 ¹	d _e	d _r	ImpAv	ImpRe	Echant	M ₂ T	M ₁ T	M ₃ SC	M ₁ SC	M ₂ M	M ₁ M	M ₃ (ASB)	M ₁ Ad	M ₁ Sgn	DeCal	M ₂ P	M ₁ P	MSC10	MSC20	MSC2	MSC3	MSC5	MSC6	MSC7	MSC4		
P1				5					0	1				1	1												0	0	0	0	0	0	0	0	
P2				2 5					1	3	1		1	0	1	0																			
				2 3								1	1	0	1	0																			
P3				5					3	2	1		1		0																				
				3								1	1		0																				
P4									2	6	0	0	0																						
P5	T T T			4					6	7							1										1	1	1	1	1	1	1	1	
P6		A		2					7	5							0	1									1	1							
	T T 1																0												1	1	1	1	1	1	
P7		A		3					5	4								1	1								1	1							
	T T																							1				1							
P10		1	4																											1	1	1	1	1	
P8		A		1					4	12								0	1								1	1							
P10		1	4																						1				1						
	T T																													1	1	1	1	1	
P9		A		4					12	13									0	1							1	1							
P10		1	4																						1				1						
	T T																													1	1	1	1	1	
P11		4		2					13	15										0	1						1								
P10		1	4																										1						
P12				1																0		1								1					
	T C 2																										1				1	1	1	1	
	T T T								15	14														0	0			1	1	1	1	1	1	1	1
P13	4 T T	2							14	10															1		1	1	1	1	1	1	1	1	
	2																																		1
P14	4 T T	1							10	11															0	1	1	1	1	1	1	1	1	1	1
	2																																		1
P15	T T T			4					11	9				1											0		1	1	1	1	1	1	1	1	1
P15	T T T			4					9	8				1														1	1	1	1	1	1	1	1
P15	T T T			4					8	0				1														1	1	1	1	1	1	1	1

Partition {EP}																										
Compteur Répartiteur					Etat des : CR0 ≠ 0 partout															Etat CR0 = 0 partout						
R4	R3	R2	R1	{R}	P1	P2	P3	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P4	P20	P21	P22	P23	P24	P25	
				0	1																					
				1		3																				
				3			2																			
				2																6						
				6				7																		
				7					5												5	5	5	5		
				5						4			4								4	4	4			
				4							12		12								12	12	12			
				12								13	13								13	13	13			
				13									15	15	15						15	15			15	
				15																	14	14	14	14	14	
				14												10					10					
				10														11			11					
				11															9							
				9															8							
				8															0							
Rappel des états de la syntaxe →																				MSC 8	MSC 4	MSC (16, 17)	MSC 3	MSC 2	MSC 20	MSC 10

$$\text{posons } A_1 = \text{MSC10} + \text{MSC20} + \text{MSC2} + \text{MSC3} + B_1$$

$$B_1 = \text{MSC5} + \text{MSC6} + \text{MSC7}$$

3°) Equations des bascules aux états du compteur et aux partitions :

a) Bascule R₁ :

$$\begin{cases} sR_1 = P_1 + P_5 + P_9 + P_{14} + 12 \cdot (P_{10} + P_{20} + P_{21} + P_{22}) \\ \quad \quad \quad + 10 \cdot P_{20} \\ rR_1 = P_3 + P_7 + 5 \cdot (P_{10} + P_{20} + P_{21} + P_{22}) \\ \quad \quad \quad + 15 \cdot (P_{20} + P_{21} + P_{22} + P_{23} + P_{24} + P_{25}) \\ \quad \quad \quad + 9 \cdot P_{15} + \text{MZG} \end{cases}$$

b) Bascule R₂ :

$$\begin{cases} sR_2 = P_2 + P_{11} + P_{12} + 13 \cdot (P_{10} + P_{20} + P_{21} + P_{24}) \\ rR_2 = P_6 + 7 \cdot (P_{20} + P_{21} + P_{22} + P_{23}) + 11 \cdot P_{15} + \text{MZG} \end{cases}$$

c) Bascule R₃ :

$$\begin{cases} sR_3 = P_4 \\ rR_3 = P_{13} + 14 \cdot P_{20} + \text{MZG} \end{cases}$$

.../...

d) Bascule R_4 :

$$\begin{cases} sR_4 = P_8 + 4 \cdot (P_{10} + P_{20} + P_{21} + P_{22}) \\ rR_4 = 8 \cdot P_{15} + MZG \end{cases}$$

4*) Equations des bascules en fonction des entrées :

Nous déterminerons toutes ces équations en fonction des équations des partitions. Nous ne tiendrons pas compte des ordres qui sont à l'état zéro (voir le principe que nous avons admis ci-dessus).

Tous les ordres figurant dans les équations des bascules doivent être précédés du symbole CR que nous n'avons pas écrit pour alléger les équations.

a) Bascule R_1 :

$$\begin{cases} sR_1 = C_1 \cdot \left[\begin{aligned} &MzT \cdot MzSC \cdot \overline{(A_1 + SC_4)} \cdot [A_2 a_1 + A_8 a_1 + A_9 a_1] \\ &+ MnSC \cdot (A_1 + SC_4) + Mz(A, S, E) \cdot [MSC10 + MSC20] \\ &+ MnP \cdot A_1 \\ &+ \overline{R_2} \cdot R_3 \cdot R_4 \cdot [\text{Décal. MSC2} + MSC4 + B_1 + MSC3] \\ &+ R_2 \cdot \overline{R_3} \cdot R_4 \cdot MSC4 \end{aligned} \right] \\ rR_1 = MZG + C_1 \cdot \left[\begin{aligned} &(\text{Imp Av} + \text{Imp Re}) \cdot \overline{MnT} \\ &+ MzM \cdot MnM (MSC10 + MSC20) \\ &+ \overline{R_2} \cdot R_3 \cdot \overline{R_4} \cdot (\text{Décal. MSC2} + MSC3 + MSC4 + B_1) \\ &+ R_2 \cdot R_3 \cdot R_4 \cdot (A_1 + MSC4) \\ &+ \overline{R_2} \cdot \overline{R_3} \cdot R_4 \cdot MzT (A_1 + MSC4) \end{aligned} \right] \end{cases}$$

b) Bascule R_2 :

$$\begin{cases} sR_2 = C_1 \cdot \left[\begin{aligned} &(\text{Imp Av} + \text{Imp Re}) \cdot MnT \\ &+ MnAd \cdot MSC10 + MSgn \cdot MSC3 \\ &+ R_1 \cdot R_3 \cdot R_4 \cdot (\text{Décal. MSC2} + MSC4 + B_1 + MSC20) \end{aligned} \right] \\ rR_2 = MZG + C_1 \cdot \left[\begin{aligned} &MzM (MSC10 + MSC20) \\ &R_1 \cdot R_3 \cdot \overline{R_4} (MSC4 + B_1 + MSC2 + MSC3) \\ &R_1 \cdot \overline{R_3} \cdot R_4 \cdot MzT (A_1 + MSC4) \end{aligned} \right] \end{cases}$$

.../...

c) Bascule R_3 :

$$\begin{cases} sR_3 = \overline{C_1} \cdot \overline{R_1} \cdot R_2 \cdot \overline{R_4} \\ rR_3 = MZG + \overline{C_1} \cdot [MzP \cdot A_1 + \overline{R_1} \cdot R_2 \cdot R_4 \cdot MSC4] \end{cases}$$

d) Bascule R_4 :

$$\begin{cases} sR_4 = \overline{C_1} \cdot [\overline{MzM} \cdot MnM (MSC10 + MSC20) \\ \quad \overline{R_1} \cdot \overline{R_2} \cdot R_3 (\text{Décal} \cdot MSC2 + MSC4 + B_1 + MSC3)] \\ rR_4 = MSG + \overline{C_1} \cdot \overline{R_1} \cdot \overline{R_2} \cdot \overline{R_3} \cdot MzT (A_1 + MSC3) \end{cases}$$

Toutes ces bascules peuvent être du type JKC ; RSC ; nous avons choisi JKC.

Les équations s'écriront alors :

$$\begin{cases} sR_i = \overline{C_1} \cdot \overline{R_1} \cdot f(s_i) \\ rR_i = \overline{C_1} \cdot R_1 \cdot f(r_i) + MZG \end{cases}$$

$f(s_i)$ et $f(r_i)$ se déterminant facilement en se reportant aux équations ci-dessus.

Il nous reste à simplifier les équations de chaque bascule, nous ne le ferons pas ici.

7-5- Conclusions :

Nous venons d'établir le principe de l'unité de lecture de un caractère. Afin d'en avoir une idée d'ensemble le lecteur se reportera à la planche 7-1 jointe, sur laquelle nous avons représenté les schémas des sous-unités établies et leurs liaisons (figurées par des "bus" orientés de la sous-unité, où est élaboré l'état, ou le signal, vers la sous-unité où ce signal est utilisé.).

La solution que nous avons choisie quant à la rétroaction des compte-rendus d'ordres, peut prêter à contestations, comme pour tous les circuits rédundants de détections partielles de pannes. Nous avons conçu un système à blocage en l'état interne du compteur répartiteur où la panne s'est produite. Nous n'avons pas envisagé tous les cas de panne (nous n'avons pas utilisé les zéros des ordres dans la table des ordres). Cela pourrait s'envisager mais coûterait encore plus cher. Nous avons encore supposé que tous les ordres étaient bouclés par un compte-rendu d'ordre afin de simplifier les notations de notre étude. En pratique, seuls les ordres émis vers des "baies" de logique différentes, seront bouclés par un compte-rendu.

8 - LES UNITES DE PROGRAMMATION

Pages :

<u>8-1- L'unité de programmation des états globaux</u>	157
8-1-1- Généralités relatives à cette machine	157
8-1-2- Etablissement du diagramme de fluence de la machine	159
8-1-3- Mise en équation en fonction des états des entrées	164
8-1-4- Equations en fonction des entrées	167
8-1-5- Conclusions.	168
<u>8-2- L'unité de programmation manuelle du mode de lecture</u>	168

Introduction

Nous avons défini chapitre 6, les boucles et les états essentiels de la machine, cela nous a permis de dégager les unités fonctionnelles.

Les principes d'étude étant identiques pour l'unité de programmation des états globaux et pour l'unité de programmation manuelle, nous ne développerons que la première unité.

Les conclusions du chapitre 6 et les ordinogrammes de la machine à réaliser (voir les planches 5-1 à 5-8 jointes) sont les seules données permettant de déterminer l'unité. L'on ne connaît pas les séquences des états des entrées lors de tous les cycles, mais seulement les variations des états d'une partie connue des entrées qui impliquent des variations des états des sorties.

Après avoir répertorié les entrées connues et les sorties, recherché les états interdits, l'on établit le graphe des séquences entrées-sorties en respectant les évolutions des ordinogrammes correspondant à la sous-unité. L'on est ensuite conduit (après avoir réduit les sommets et rendu les chemins déterministes) au diagramme de fluence et à la table de fluence, connus de par les chemins ou les partitions des entrées impliquant des évolutions. La mise en équation de l'unité, variable interne par variable interne s'effectue en respectant les propriétés des fonctions mémoires et des fonctions bascules.

8-1- L'unité de programmation des états globaux :

Le graphe des évolutions globales (figure 3, chapitre 6) et les ordinogrammes nous permettent d'établir l'unité logique de programmation des états globaux.

En effet, nous connaissons les entrées, les sorties et les séquences des entrées conduisent aux états des sorties dans chaque cas d'évolution globale.

8-1-1- Généralités relatives à cette unité :

Nous n'allons pas rappeler les conclusions du chapitre 6, mais nous allons préciser les entrées, les sorties, les états interdits des entrées et des sorties.

1°) Tableau des entrées et des sorties :

Signaux des entrées			
Actions manuelles	signaux mémoire rupture de programme.	signaux de synthèse générale	Mise aux C.I.
BP PL BP MPB BP UA BP USA BP BP	M M02 M M10 M M30	UT LC LT UC	MZG
Signaux ou états des sorties			
Signaux de programmation globale.		Signaux de mise à zéro des mémoires de rupture programme	
PL MPB USA UA BP		MZ MD M02 MZ MD M10 MZ MD M30	

Nous aurions pu aussi élaborer directement des signaux correspondant aux états de la programmation globale pour les différentes phases que nous avons établies au chapitre 6 (voir 6-3-1 2°). C'est à dire 9 signaux (un par phase globale). Nous avons choisi la solution des signaux (PL, MPB,

.../...

USA, UA, BP) après avoir remarqué qu'il faut élaborer ces signaux pour commander les voyants situés sur le pupitre.

2°) Les états interdits des entrées et des sorties :

a) les entrées :

。 Pour toutes les actions manuelles (BP PL ... BP BP) l'opérateur peut appuyer sur n'importe quel bouton poussoir à n'importe quel instant, suivant une séquence absolument quelconque. Nous n'avons pas d'états interdits pour ce sous-ensemble d'entrées.

。 Pour les signaux de synthèse générale : UT, LC, LT, UT un seul signal peut être à l'état 1, ou aucun n'est à l'état 1. Tous les autres sont des états interdits.

。 Pour les états mémoires de rupture de programme :

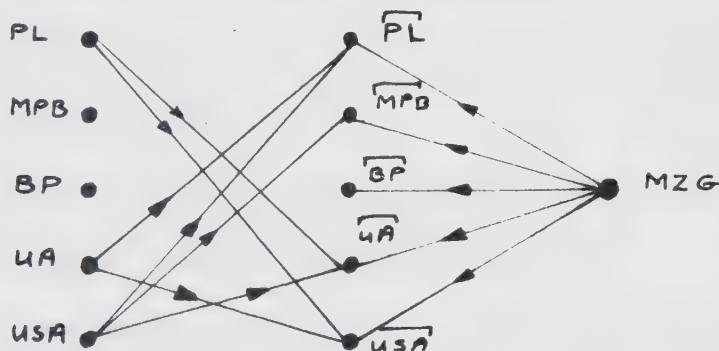
Le registre mémoire de rupture de programme est un registre particulier où nous avons utilisé 4 variables internes afin d'avoir des évolutions consécutives entre tous les états

Nous avons 5 états possibles, mais de par la conception du registre aucun état n'est interdit.

b) Les sorties :

Reprenons les résultats de la conclusion (6-3-2°).

Nous allons représenter les états interdits par un graphe des états des signaux interdits qui est plus pratique et plus clair que d'écrire tous les états interdits. Nous ajouterons les états des sorties impliqués par MZG.



Ce graphe se lit : $MZG \rightarrow \overline{PL} \cdot \overline{MPB} \cdot \overline{BP} \cdot \overline{UA} \cdot \overline{USA}$

.../...

8-1-2- Etablissement du diagramme de fluence de la machine et la table d'évolution.

Nous avons à établir une machine synchrone par rapport à UT ou LT, en effet nous avons vu que les changements d'états des sorties ne se produisent que si UT ou LT sont à l'état 1. Le cas de la mise aux conditions initiales forcées par MZG étant une évolution forcée pouvant être non consécutive, ne pose pas de problème.

Nous pouvons appliquer par exemple les méthodes classiques de GINSBURG ou d'AIZERMAN permettant d'établir la table de fluence de la machine. Mais avant il est nécessaire de déterminer la séquence des états des sorties en fonction de la séquence des états des entrées.

Plutôt que d'essayer d'écrire ces séquences directement, nous allons établir le graphe des séquences des sorties en fonction des séquences d'états des entrées.

Puis nous déterminerons le diagramme de fluence (graphe des évolutions d'état interne) de cette machine.

1°) Graphe des séquences entrée-sortie :

(Voir figure 1 page :162)

En fonction du graphe des évolutions globales (figure 3 chapitre 6-1-3), des états globaux associés à chaque phase et des ordinogrammes $P_1(M01)$ et $P_3(M01)$ nous allons tracer le graphe des séquences des états des sorties en fonction des états des entrées.

A chaque sommet nous indiquerons uniquement les sorties qui changent d'état par rapport à leur état précédent. (Nous indiquerons l'état qu'elle peut prendre sous l'influence de l'état des entrées).

Chaque chemin sera repéré par l'état des entrées qui change d'état par rapport à l'état initial des entrées, l'on n'indiquera pas les chemins bouclés).

L'on commencera par l'état initial impliqué par MZG et l'on décrira toutes les séquences prévues permettant de décrire les phases d'évolutions des ordinogrammes en respectant la programmation des états globaux.

En parallèle, au fur et à mesure que l'on trace le graphe, l'on établit la table des états des sorties (figure 2), pour chaque sommet du graphe des séquences. Cette table est

.../...

complétée par les états aléatoires possibles dans le cas où sur le graphe l'on a prévu que plusieurs signaux de sortie pouvaient changer d'état.

2°) Recherche des sommets identiques du graphe des séquences (états identiques des sorties).

Pour chaque sommet l'on détermine la valeur octaire inverse associée à chaque état du tableau. Plutôt que de garder la numérotation octaire, nous avons affecté par état un numéro. Nous avons ainsi les sommets de 1 à 14, plus les sommets 20 à 22 qui sont des sommets de transitions possibles.

Nous avons déterminé ainsi les états de la séquence des sorties.

3°) Recherches des états des entrées impliquant des changements d'état des sorties : (Figures 2;3 ,voir page 163)

L'on établit la table des états des entrées (figure 3) pour tous les chemins du graphe des séquences. Les chemins ne sont pas indexés par le numéro du sommet initial et le numéro du sommet final du graphe des séquences mais par les numéros des sommets identiques (états des sorties déterminés figure 2).

Cette table n'est pas complètement définie, nous avons précisé uniquement les actions manuelles qui impliquent une évolution des sorties sans nous occuper de l'état des autres boutons poussoirs, que nous supposons momentanément à l'état zéro.

L'on détermine les états identiques en établissant la valeur octaire inverse associée aux états des entrées. L'on détermine ainsi 14 états des entrées notés de P_1 à P_{14} qui sont des états d'évolution de la machine.

Nous allons lever les indéterminations quant à l'état des boutons poussoirs par adjonction de sécurité. Nous avons la première solution simple qui consiste à imposer l'état zéro pour tous ceux qui ne sont pas prévus à l'état un (solution trop coûteuse). Nous allons rendre déterministe les évolutions d'état des sorties en considérant tous les chemins émergents des sommets identiques.

a) toutes les actions manuelles, BP BP exclues, sont synchrones par rapport à UT, et BP BP est synchrone par rapport à LT ; $UT.LT = 0$ aucun problème ne se pose pour BP BP.

.../...

b) Pour le sommet ① : $E_{1-2} \cap E_{1-14} \neq \emptyset$, il faut ajouter $\overline{BP\ USA}$ en E_{1-2} dans ce cas E_{1-14} est prioritaire par rapport à E_{1-2} .

c) Pour le sommet ⑤ : $E_{5-6} \cap E_{5-7} \neq \emptyset$, nous ajouterons $\overline{BP\ UA}$ en E_{5-6} et $\overline{BP\ UA}$ en E_{5-7} (P_4) aucune évolution n'est prioritaire.

d) Il nous reste : l'évolution E_{3-1} qui transite par l'état s_2 des sorties, ajoutons la sécurité classique telle que $E_{2-3} \cap E_{3-1} = \emptyset$ en rendant E_{3-1} prioritaire par rapport à E_{2-3} , c'est à dire en élaborant :

$$BP\ MPB \cdot \overline{BP\ UA} \cdot \overline{BP\ USA}.$$

Pour l'évolution E_{7-1} qui transite soit par s_5 soit par s_7 : $E_{7-1} \cap E_{5-7} = \emptyset$; $E_{7-1} \cap E_{9-10} = \emptyset$

$$E_{7-1} \cap E_{5-6} = \emptyset$$

e) Conclusions :

Les signaux BP PL, BP MPB, BP UA, BP USA, BP BP étant des sorties de contacts électromécaniques il est nécessaire d'élaborer des signaux logiques électroniques dans l'interface (électrotechnique, logique électronique) nous élaborons directement.

$$\begin{aligned}\overline{PL} &= BP\ PL \cdot \overline{BP\ UA} \cdot \overline{BP\ USA} \\ \overline{MPB} &= BP\ MPB \cdot \overline{BP\ UA} \cdot \overline{BP\ USA} \\ \overline{UA} &= BP\ UA \cdot \overline{BP\ USA} \\ \overline{USA} &= BP\ USA \cdot \overline{BP\ UA} \\ \overline{BP} &= BP\ BP\end{aligned}$$

Si nous tenons compte des états interdits pour UT, LT et pour M M02, M M10, M M30, les différents états des entrées P_i s'écrivent alors de la manière suivante :

$$\begin{aligned}P_1 &= UT \cdot \overline{PL} \cdot A \quad \text{avec } A = \overline{M\ M02} \cdot \overline{M\ M10} \cdot \overline{M\ M30} \\ P_2 &= UT \cdot \overline{MPB} \cdot A \\ P_3 &= LT \cdot \overline{BP} \cdot A \\ P_9 &= LT \cdot \overline{BP} \cdot M\ M30 \\ &\text{etc...}\end{aligned}$$

Nous avons ainsi élaborer des états $P_1, P_2, \dots$ tels que toutes évolutions sur pannes des entrées soient possibles, solution économique mais pouvant conduire en cas de panne de la logique à des évolutions non prévues.

.../...

Recherche des sommets identiques													
Sorties Etats				Sommets graphe séquences	10 ¹			10 ²		10 ³			
	10 ³	10 ²	10 ¹		PL	MPB	BP	UA	USA	MZMOM2	MZMDM10	MZMDM30	
Etats des sommets	1	0	0	0	1	0	0	0	0	0	0	0	0
	2			4	2	1	0	0	0	0	0	0	0
	3			5	3	1	1	0	0	0	0	0	0
	4			7	4	1	1	1	0	0	0	0	0
	5		1	1	5-8	0	0	1	0	0	0	0	0
	6		1	1	6	0	0	1	0	1	0	0	0
	5		1	1	7	0	0	1	0	0	0	0	0
	7		2	1	9	0	0	1	1	0	0	0	0
	5			1	10	0	0	1	0	0	0	0	0
	6		1	1	11	0	0	1	0	1	0	0	0
	8	2	1	1	12	0	0	1	0	1	0	1	0
	6		1	1	13	0	0	1	0	1	0	0	0
	5			1	14	0	0	1	0	0	0	0	0
	9		2		15	0	0	0	1	0	0	0	0
	10		2	2	16	0	1	0	1	0	0	0	0
	11	1	2	2	17	0	1	0	1	0	0	0	1
	10		2	2	18	0	1	0	1	0	0	0	0
	12		2	3	19	0	1	1	1	0	0	0	0
	7		2	1	20	0	0	1	1	0	0	0	0
	1				21	0	0	0	0	0	0	0	0
13	4			22	0	0	0	0	0	1	0	0	
1				23	0	0	0	0	0	0	0	0	
1				24	0	0	0	0	0	0	0	0	
14	1			25	0	0	0	0	1	0	0	0	
Etats de Transitions	20		3	5 ₁	0	1	1	0	0	0	0	0	0
	3		5	5 ₂	1	0	1	0	0	0	0	0	0
	5		2	21 ₁	0	0	0	1	0	0	0	0	0
	5		1	21 ₂	0	0	1	0	0	0	0	0	0
	21		2	24 ₁	0	1	0	0	0	0	0	0	0
22		4		24 ₂	1	0	0	0	0	0	0	0	

Table des sorties

Figure 2

Entrées - Chemins.																		
Partition {P}	Valeur octaire Inverse				chemins de a _i		10 ¹				10 ²		10 ³		10 ⁴			
							UT	LT	BPPL	BPMPB	BPBP	BP UA	BP USA	MDM02	MDM10	MDM30		
	10 ⁴	10 ³	10 ²	10 ¹														
1	0	0	4	2	1	2	1	0	1	-	-	0	0	0	0	0	0	0
4		1		2	1	14	1	0	-	-	-	0	1	0	0	0	0	0
10	4		4	2	1	13	1	0	1	-	-	0	0	1	-	-	-	-
11	4	1		2	1	13	1	0	-	-	-	0	1	1	-	-	-	-
2			2	2	2	3	1	0	-	1	-	0	0	0	0	0	0	0
4		1		2	3	1	1	0	-	-	-	0	1	0	0	0	0	0
3			1	1	3	4	0	1	-	-	1	-	-	0	0	0	0	0
5		2		2	4	5	1	0	-	-	-	1	0	0	0	0	0	0
4		1		2	4	5	1	0	-	-	-	0	1	0	0	0	0	0
4		1		2	5	6	1	0	-	-	-	0	1	0	0	0	0	0
6	2	1		2	5	6	1	0	-	-	-	0	1	0	1	0	0	0
5		2		2	5	7	1	0	-	-	-	1	0	0	0	0	0	0
5		2		2	6	5	1	0	-	-	-	1	0	0	0	0	0	0
7	2	2		2	6	8	1	0	-	-	-	1	0	0	1	0	0	0
12	2			2	7	5	1	0	-	-	-	-	-	0	1	0	0	0
14	4			2	7	1	1	0	-	-	-	-	-	1	-	-	-	-
13	1			2	7	9	1	0	-	-	-	-	-	0	-	1	0	0
5		2		2	8	6	1	0	-	-	-	1	0	0	0	0	0	0
8	1		2	2	9	10	1	0	-	1	-	0	0	0	0	0	1	0
9	1		1	1	10	11	0	1	-	-	1	-	-	0	0	0	1	0
3			1	1	10	12	0	1	-	-	1	-	-	0	0	0	0	0
3			1	1	11	10	0	1	-	-	1	-	-	0	0	0	0	0
5		2		2	12	7	1	0	-	-	-	1	0	0	0	0	0	0
4		1		2	13	1	1	0	-	-	-	0	1	0	0	0	0	0
1			4	2	13	1	1	0	1	-	-	0	0	0	0	0	0	0
1			4	2	14	1	1	0	1	-	-	0	0	0	0	0	0	0

Table des entrées

Figure 3

4°) Diagramme de fluence et table : (Figures 4;5;voir page165)

Considérons la table des entrées figure 2 obtenue après adjonction des sécurités. Nous remarquerons les sorties de la machine, étant à deux époques t_1 et t_2 différentes, dans un même état si $\in [s_1 \dots s_{14} \dots s_{20} \dots s_{22}]$ quelconque, il n'existe pas de séquence des états des entrées telles que à $t_1 + \Delta t$ les sorties prennent l'état s_j et que à $t_2 + \Delta t$ les sorties prennent (par la même séquence) l'état $s_1 \neq s_j$. A chaque état stable des sorties s_i nous pouvons associer un état interne q_i de la machine.

L'on détermine alors le diagramme de fluence de la figure 4 ainsi que la table de fluence de la figure 5.

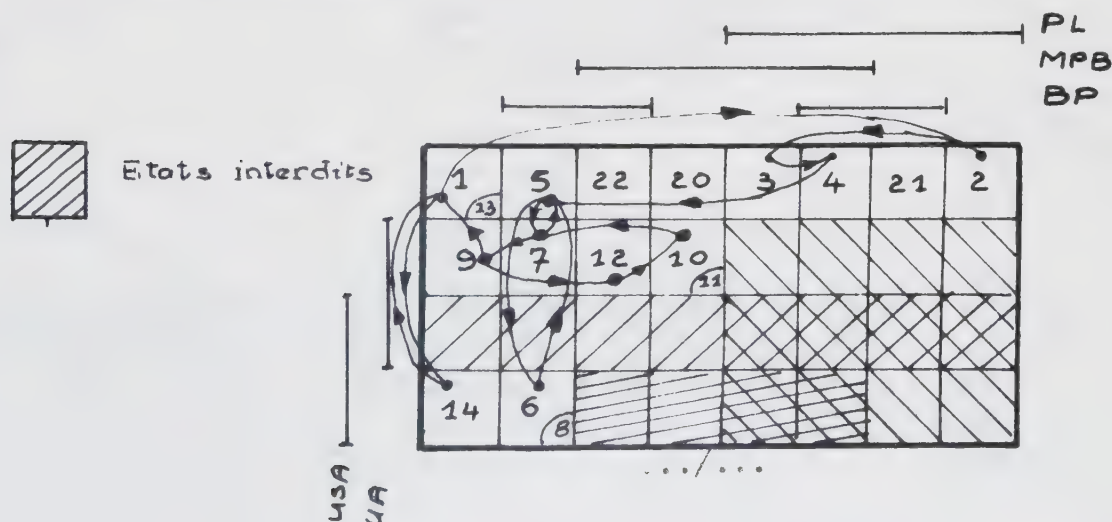
En plus MZ M10, MZ M30, MZ M02 sont tels que, par exemple $MZ M10 \rightarrow M M10$; les sorties MZ M10, MZ M20, MZ M02 seront du type MEALY. Nous réduirons la table de fluence figure 5 en supprimant les états des entrées P_7, P_9, P_{10}, P_{11} et les lignes 8, 11, 13. Il nous reste alors les états stables correspondant à des états stables des sorties suivant $s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_9, s_{10}, s_{12}, s_{14}$, et éventuellement les états de transition 20, 21, 22.

8-1-3- Mise en équation en fonction des états des entrées :

1°) Codage :

Nous ne voulons pas d'autres états des sorties que ceux prévus dans la table des états des sorties, dans ce cas, toutes les évolutions des sorties de la machine seront consécutives, plutôt que de chercher un codage particulier des variables internes, nous allons choisir PL, MPB, BP, UA comme variables internes.

2°) Tableau des états possibles des sorties (choisies comme états internes)



Etats des Sorties							
M3M M30	M3M M10	M3M M02	USA	UA	BP	MPB	PL
							1
					1	1	1
			1		1		
				1	1		
1		1		1			
				1			
					1		
1				1		1	
					1	1	1
		1					
			1				
						1	
					1		1
</							

Figure 5

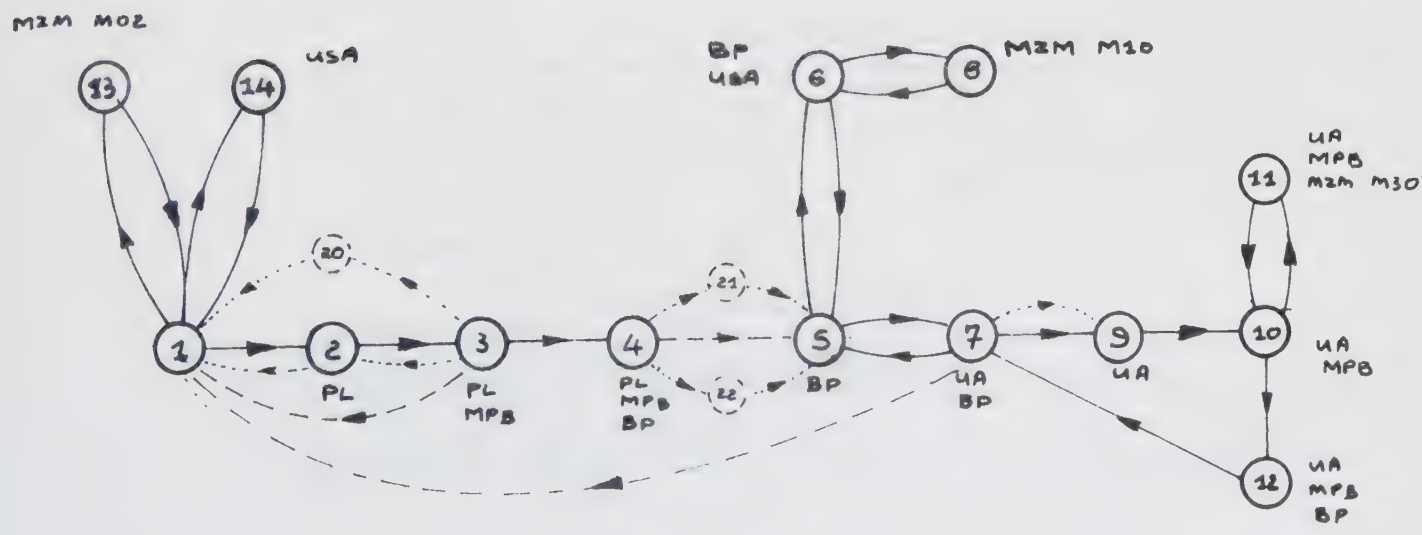


Figure 4

6°) Mise en équation des sorties MZ M10, MZ M30, MZ M02 :

En tenant compte des états interdits :

$$MZ M10 = USA \cdot BP \cdot P_7$$

$$MZ M30 = UA \cdot MPB \cdot P_9$$

$$MZ M02 = \overline{UA} \cdot \overline{BP} \cdot \overline{PL} \cdot \overline{MPB} \cdot \overline{USA} \cdot (P_{10} + P_{11})$$

8-1-4- Equation en fonction des entrées :

Nous n'allons pas reprendre toutes les équations, considérons simplement l'équation de PL.

$$PL = \overline{MZG} \cdot \overline{UA} \cdot \overline{USA} \cdot (P_4 + BP \cdot P_5) \cdot (\overline{MPB} \cdot \overline{BP} \cdot P_1 + PL)$$

avec :

$$P_1 = \overline{UT} \cdot A \cdot \overline{PL}$$

$$P_4 = \overline{UT} \cdot A \cdot \overline{USA}$$

$$P_5 = \overline{UT} \cdot A \cdot \overline{UA}$$

1°) Etudions les chemins de cet opérateur PL

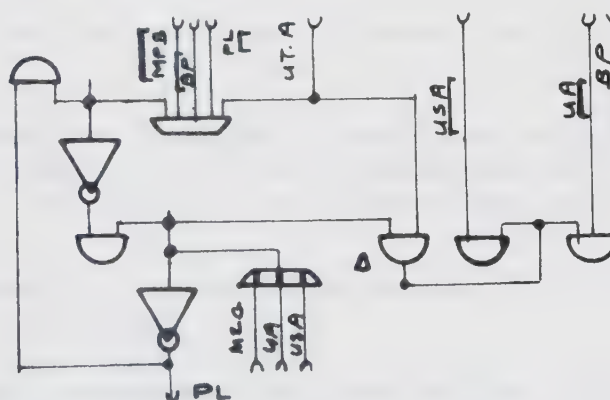
◦ MZG + UA + USA est un chemin forcé pour PL (mise à zéro prioritaire)

◦ $(P_4 + BP \cdot P_5) \cdot (\overline{MPB} \cdot \overline{BP} \cdot P_1) = \emptyset$
avec en plus $\overline{UT} \cdot A$ le signal d'horloge.

2°) Opérateur :

Nous pourrions utiliser tout opérateur mémoire ou bascule à niveau du type R S C ayant en plus une entrée MZ de mise à zéro prioritaire.

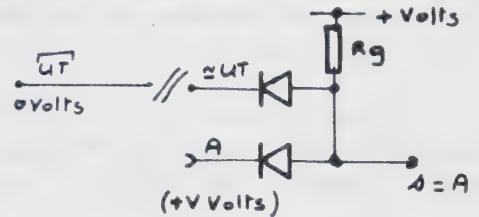
Réalise avec la logique CII [16] le schéma suivant a été retenu.



8-1-5- Conclusions :

Nous avons établi une machine à évolution sur panne des entrées, car nous avons affecté les états interdits des entrées à la plupart des chemins afin de diminuer le nombre de signaux intervenant dans les équations. Nous avons ainsi réalisé une économie importante dans les circuits combinatoires d'élaborations des signaux de mise à un et de mise à zéro des mémoires, gagné aussi sur le décodage (en choisissant le codage des sorties et non des variables internes particulières).

Considérons une panne classique des entrées, par exemple le fil de liaison entre l'opérateur qui élabore $\overline{UT}$ et l'unité de programmation, est déconnecté (en l'air). Le signal $\overline{UT}$ vu par l'unité de programmation est toujours à l'état UT, jamais à l'état $\overline{UT}$ (du fait de la technologie de la logique). Toutes les fonctions prévues avec l'état $\overline{UT} = 1$ seront correctes, restent celles avec $UT = 1$, nous aurons simultanément dans l'unité l'état $\overline{UT} = 1$ et un état UT équivalent à 1 d'où des fonctionnements anormaux. (Exemple évolutions $\textcircled{3} \rightarrow \textcircled{1}$ et $\textcircled{3} \rightarrow \textcircled{4}$ possibles au synchronisme avec LT et le faux signal $\overline{UT}$).



Le remède est simple, il faut que le signal reliant les deux unités soit UT et que l'on élabore $\overline{UT}$ dans l'unité de programmation, l'on a ainsi augmenté la fiabilité de la machine. L'on peut améliorer encore en élaborant $UT' = UT \cdot \overline{LT}$ dans l'unité mais nous aurons toujours des risques d'évolutions sur panne par ces méthodes. Il ne reste plus que la détection de panne.

8-2- L'unité de programmation manuelle du mode de lecture :

Les méthodes suivies pour établir cette unité sont identiques à celles de l'unité précédente, les séquences de fonctionnement étant nombreuses (confère les boucles des ordinogrammes SP1 (M01); SP6 (M01) ; SP7 (M01) ; SP2 (M01) nous n'aborderons pas l'étude détaillée de cette unité dans cette thèse (les graphes initiaux comportant 120 chemins).

Nous préciserons seulement les familles du graphe des séquences entrées-sorties que nous avons été amenés à étudier afin de concevoir et réaliser cette unité.

Nous avons deux familles de fonctionnement impliquant deux familles de graphe.

1°) Le fonctionnement en usinage automatique normal (voir le paragraphe 6-2-2 page 117 et les ordinogrammes SP1(M01) et SP2(M01)), conduit au premier graphe dont les états des sorties sont :

- lecture manuelle en pas à pas ou en bloc à bloc
- lecture automatique.

Ces états sont programmés par les boutons poussoirs correspondant, en synchronisme par rapport à LcaT (lecture caractère terminée) c'est à dire CR s2d1 ou CR₃ élaborés par l'unité de lecture de un caractère.

L'ordre impulsion d'usinage impliquant si la programmation est correcte l'ordre A₂a₁ de lecture de un caractère.

.../...

2°) Le fonctionnement en mise en place bande (voir le paragraphe 6-2-3 page 118 et les ordinogrammes SP6(M01) SP7(M01)) conduit au deuxième graphe dont les états des sorties sont :

- lecture caractère à caractère en avance ou en retour
- lecture pièce à pièce en avance ou en retour.

Ces états sont programmés par les boutons poussoirs correspondant, en synchronisme par rapport à LcaT (lecture caractère terminée) c'est à dire CRs8c1, CRs8d1 en Avance bande et CRs9c1, CRs9d1 en Retour bande.

Les ordres d'impulsions d'Avance ou de Retour impliquant respectivement si la programmation est correcte, les ordres A_{8a_1} , A_{9a_1} .

La réduction de chaque famille de graphe conduit au diagramme des séquences de la machine puis aux équations logiques.

9 - CONCLUSIONS

Quelle que soit la machine logique industrielle que l'on souhaite concevoir, le problème essentiel est d'établir ce que nous appelons l'énoncé. Le software est le problème le plus délicat, le plus vague et le plus important.

Dans la plupart des automatismes logiques industriels, l'on ne dispose même pas d'un langage. Il est nécessaire de créer de toute pièce les évolutions de la machine et leurs enchainements en fonction de la physique du système et des impératifs du client.

Les méthodes classiques d'organisation du travail (méthode PERT par exemple), conduisant à l'établissement successif des ordinogrammes globaux, puis des ordinogrammes de plus en plus détaillés est à notre avis la meilleure solution pour établir l'énoncé.

Les principaux avantages sont les suivants :

- L'énoncé est rédigé sous une forme synthétique claire, qui permet au client de le vérifier rapidement sans se perdre dans les détails, et de proposer éventuellement des modifications ou des adjonctions.

- Une fois les ordinogrammes globaux déterminés, "l'ingénieur système logique" peut "ventiler" parmi les spécialistes de l'équipe, le développement des phases d'évolutions globales et faire établir progressivement les ordinogrammes développés. Il répartit ainsi les tâches en fonction des possibilités de chacun.

Malheureusement comme pour tous les problèmes de software pour lesquels l'on ne dispose pas d'une formulation mathématique, ni d'un langage précis, les méthodes formelles pratiques d'étude sont encore en grande partie à déterminer.

Il ne semble pas qu'il soit possible d'établir des algorithmes précis, cela imposerait alors que l'on ne connait rien, de tenir compte de tout. L'on est conduit à l'idée, maintenant classique des algorithmes flous, permettant de guider les techniciens lors du développement des ordinogrammes, et de les vérifier au fur et à mesure du développement de l'énoncé.

。 L'ordinogramme permet de définir des sous-ensembles de complexité moindre auxquels correspondent en logique câblée, des unités fonctionnelles dont il sera possible de répartir l'étude à une équipe.

Une fois ces unités fonctionnelles établies, il faut déterminer les machines logiques câblées correspondantes. Malheureusement, si l'établissement direct de la table des phases puis son codage est aisé pour les machines simples, il n'en est pas de même pour les machines industrielles. L'écriture directe de la table est pratiquement impossible. Nous nous sommes orientés vers deux solutions non minimales :

。 La première solution appliquée pour l'unité fonctionnelle de lecture de un caractère consiste à fusionner toutes les phases d'évolution de même application sémantique en une seule, à établir les équations des chemins d'évolutions puis à effectuer le synthèse en utilisant un compteur répartiteur synchrone. Cette solution voisine du compteur ordinal est adaptée aux machines en réseau. Elle complète le schéma des séquences qui lui, n'est applicable pratiquement que pour les machines à évolutions en chaîne.

. La deuxième solution conduit au diagramme de fluence (connu uniquement de par ses chemins émergents et incidents) puis à la table des phases permettant d'établir les équations de la machine. Cette méthode est applicable aux machines séquentielles en général, et aux machines séquentielles industrielles en particulier. Elle a l'avantage de n'introduire dans l'étude que les états des entrées (14 états dans le cas de l'unité de programmation sur 2^{13} états au total), impliquant des évolutions à partir de chaque état interne généré, sans tenir compte des états correspondants aux chemins bouclés (stables).

Les méthodes de développement des ordinogrammes de recherche des unités fonctionnelles, de réduction des graphes des séquences entrées-sorties sont à notre avis très efficaces; elles permettent de poser pas à pas l'énoncé de toutes les évolutions de la machine, de la scinder en sous-machines, sans se perdre dans le détail des nombreux signaux logiques qu'elle devra comporter. Elles complètent les méthodes usuelles utilisées en logique binaire qui ne sont utilisables qu'à partir des diagrammes de fluences ou des tables des phases, comportant en général un nombre peu élevé d'entrées et de sorties.

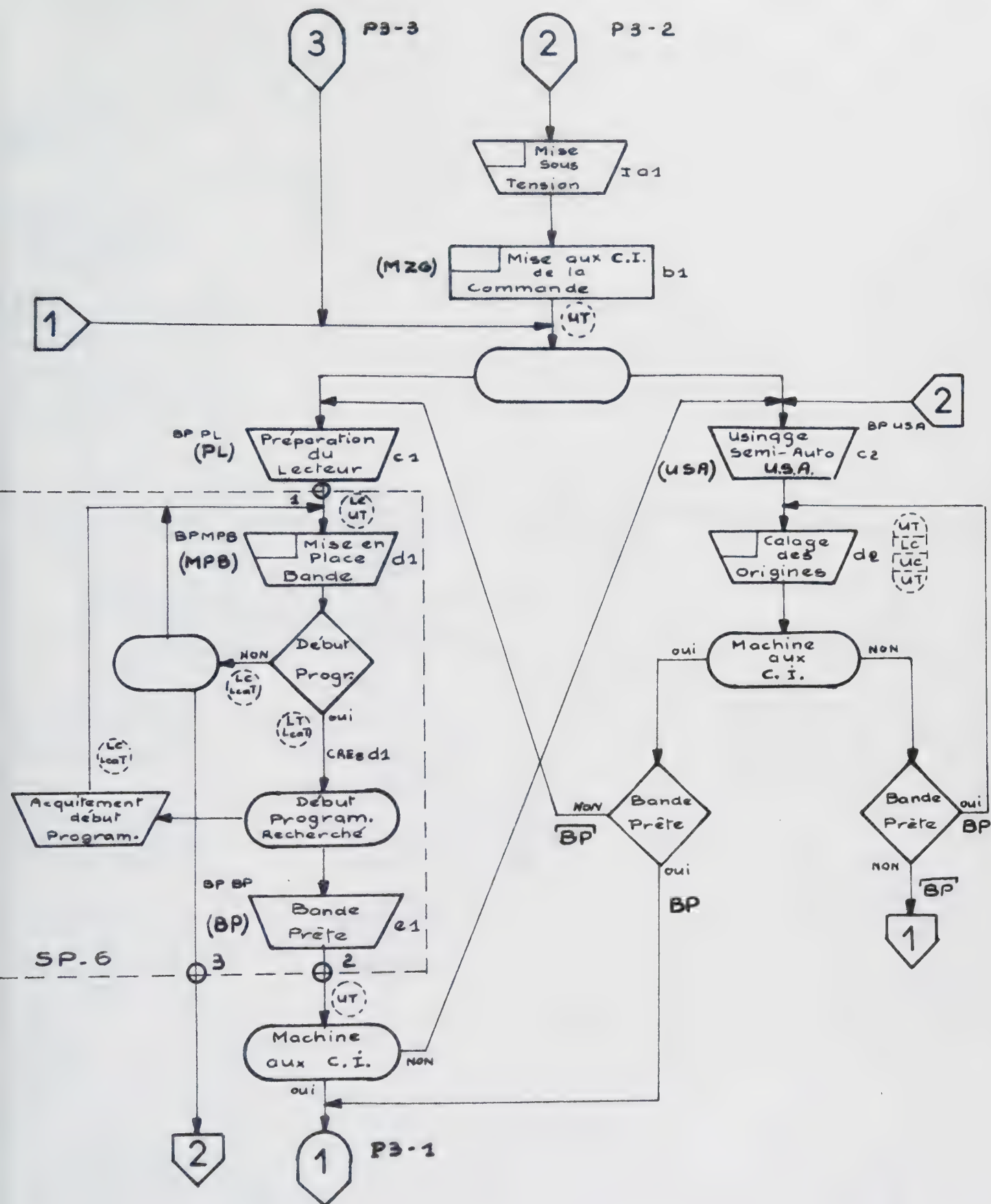
		LANGAGE	PARLÉ	Langage M.O.	
				I.S.O.	E.I.A.
< Adresses >	< Adresses Non cotées >	Numéro de Bloc, dit: Normal De début de pièce Répétitif		N % :	
		Fonctions Préparatoires Vitesse d'Avance Vitesse de Rotation de Broche Fonctions Numéro d'outil Fonctions Auxiliaires		G F S T M	
	< Adresses cotées >	Mouv. ^{ts} de Translation	Axes principaux X,Y,Z Axes Secondaires // $\vec{a}$ X,Y,Z Axes Tertiaires // $\vec{a}$ X,Y,Z	X Y Z U V W P Q R	
		Mouv. ^{ts} de Rotation	Autour des axes X;Y;Z Autour d'axes spéciaux	A B C D E	
Paramètres auxiliaires pour contournage		I J K			
< Chiffres >	< Signes >	Sens de déplacement positif { ou signes de Sens de déplacement négatif { Valeurs algébriques		+ -	
< Chiffres >		chiffres décimaux classiques		0 à 9	
< Caractères auxiliaires >	Pas de perforation dit Blanc noté: <BL> Bloc optionnel			NUL /	NUL
	Fin de Bloc, doit { Interligne Retour à la ligne } noté: <FB> Les caractères qui suivent sont { Hors programme EN programme			LF NL ()	

Langage Machine - outil		Langage parlé
Familles de mots	mots	
< Fonction auxiliaire programme >	< Arrêt machine >	M 01 Arrêt Facultatif Ξ Stop optionnel M 03 Arrêt NON Facultatif Ξ Stop NON option.
	< Rupture programme >	M 00 Arrêt Programmé M 02 Fin de programme Ξ Fin de pièce M 30 Fin de bande M 12 Démarrage du programme M 10 Usinage Semi-automatique après ce Bloc.
< Préparation machine >	< Serrage pièce >	M 68 Bridage pièce M 69 Débridage pièce M 60 changement de pièce
	< Serrage Table >	M 78 Bridage table M 79 Débridage table
		M 10 Bridage M 11 Débridage
	< Arrosage >	M 08 Arrosage N°1 marche M 07 Arrosage N°2 marche M 09 Arrêt arrosage M 50 Arrosage N°3 marche M 51 " " N°4 " "
< Préparation broche >	< Rotation broche >	M 06 Changement d'outil M 19 Arrêt broche orientation déterminée M 03 Rotation broche sens Anti-Trigo M 04 " " " " Trigonométrique M 05 Arrêt de la rotation broche M 13 Rotation broche sens A.T. + arrosage M 14 " " " " Trigo + " "
	< Gamme de vitesse broche >	M 38 Gamme de vitesse de rotation N°1 M 39 " " " " N°2

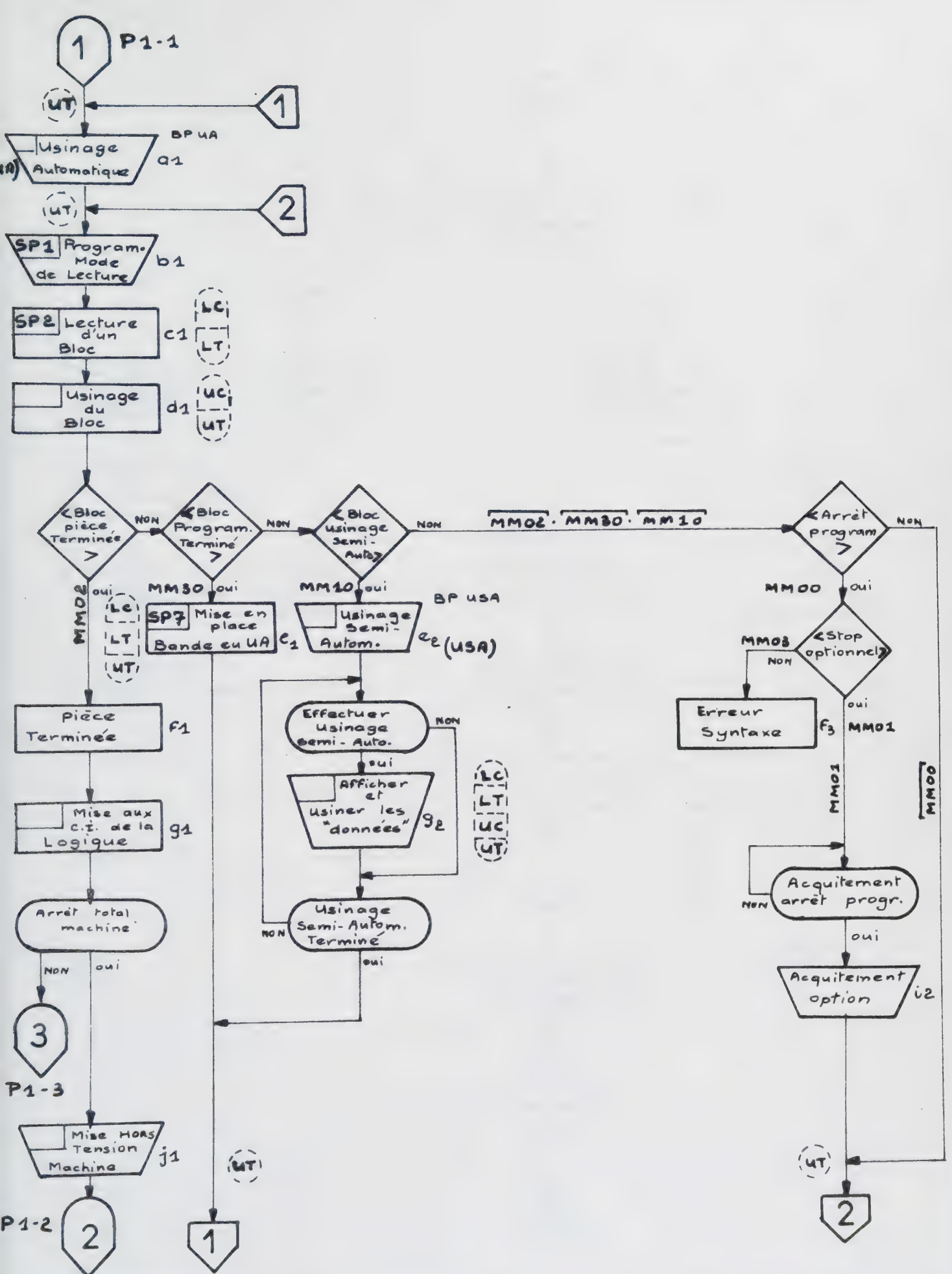
Mots Fonctions Auxiliaires

Familles et applications sémantiques

PLANCHE N°4-2

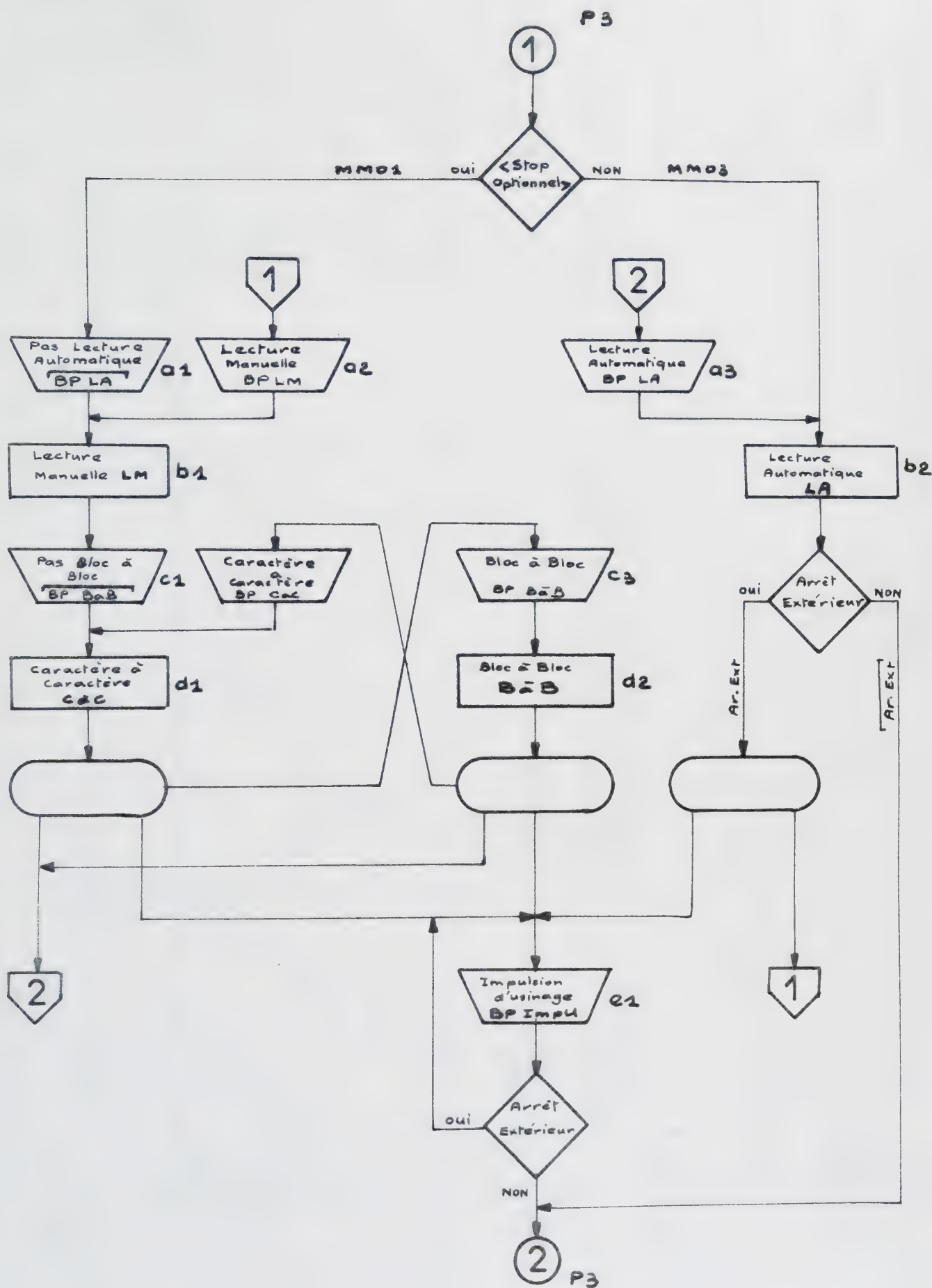


Ordinogramme global
 Démarrage - Préparation du lecteur
 Mise aux origines des axes de la machine



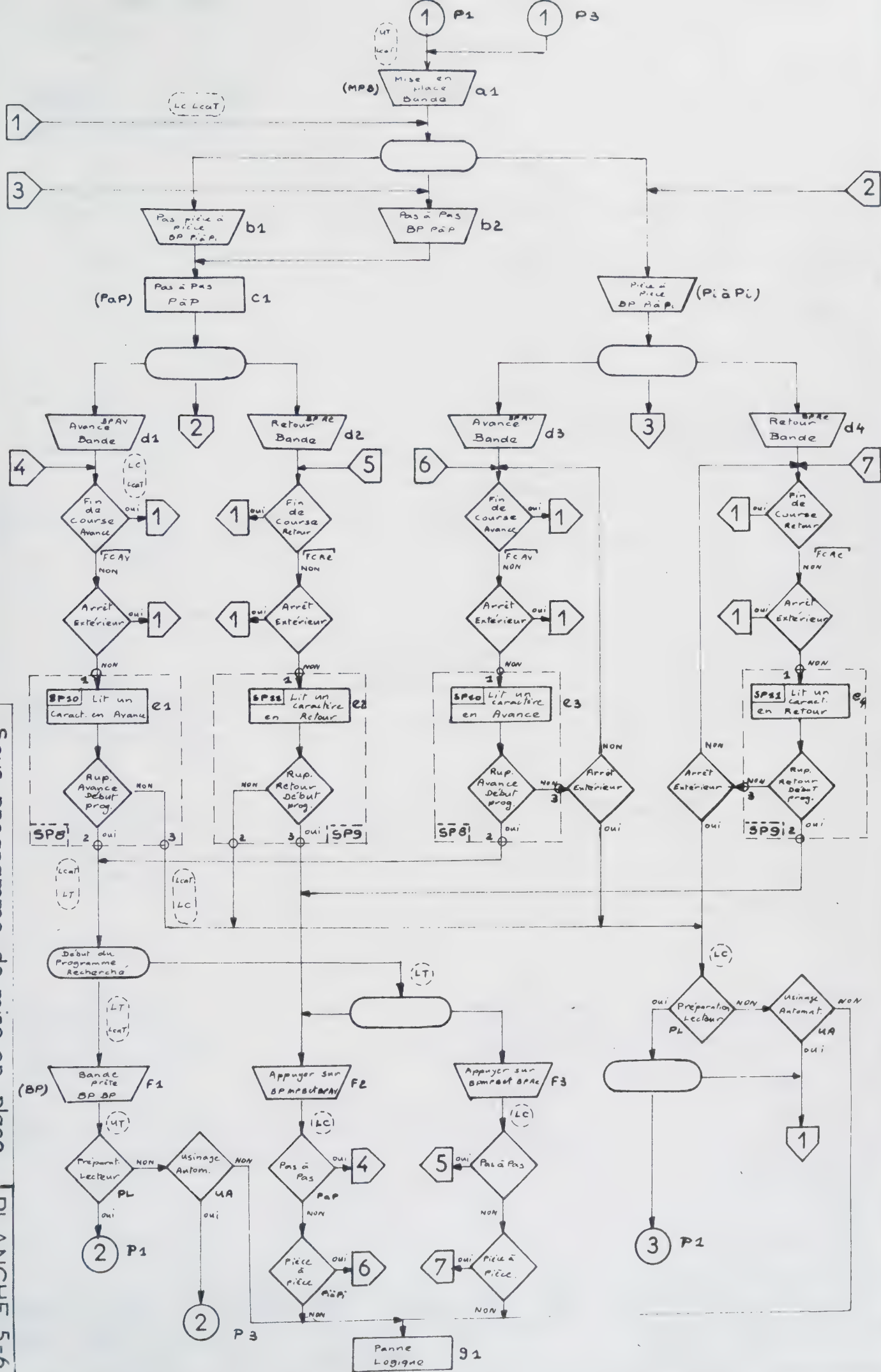
Ordinogramme global
des rupture de séquences en fin
d'usinage d'un bloc

PLANCHE 5-2
Ordinogramme
P^{III}- MO1



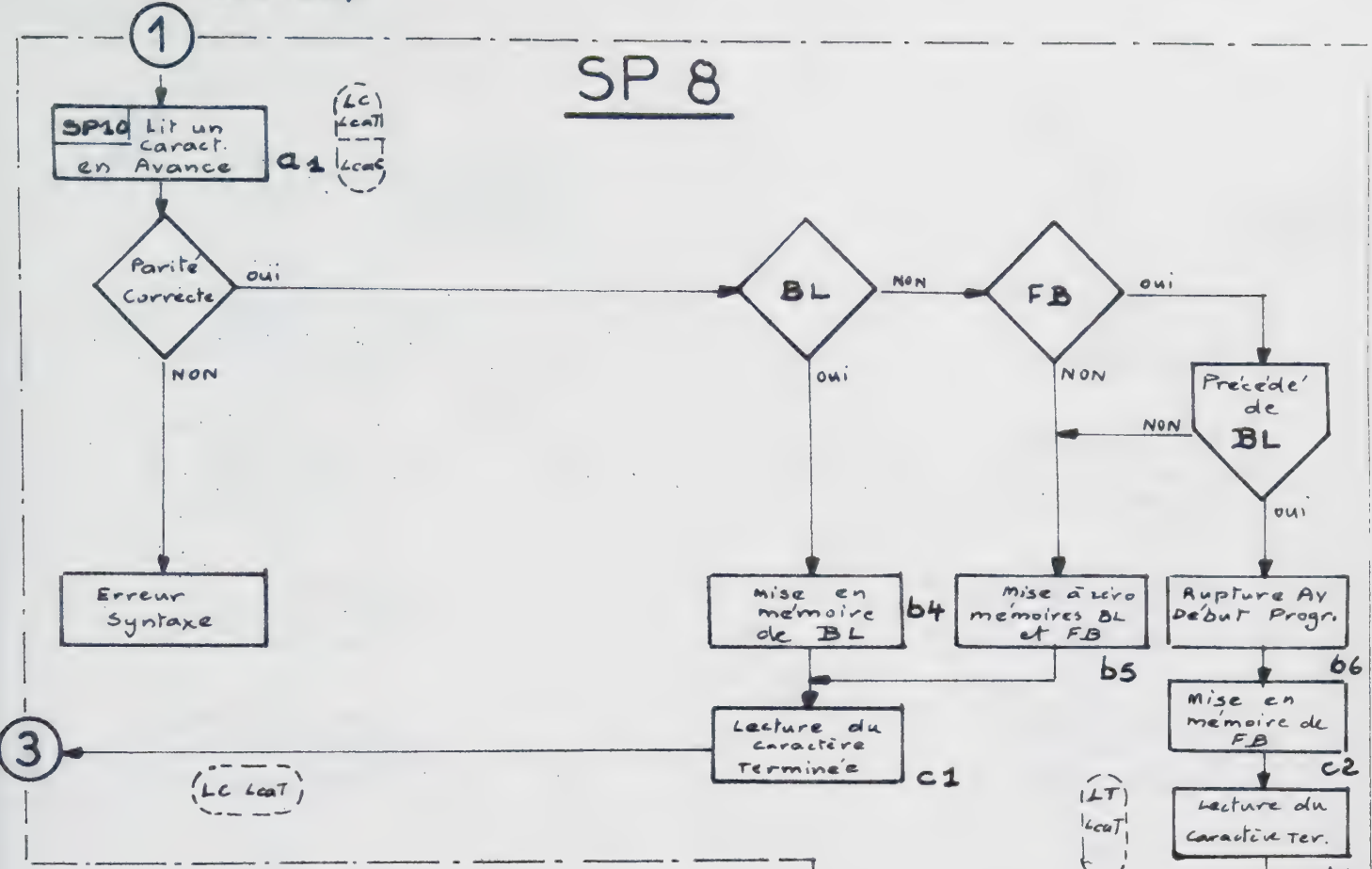
Ordinogramme de la programmation manuelle en usinage automatique

PLANCHE 5-3
Ordinogramme
SP1-MO-1



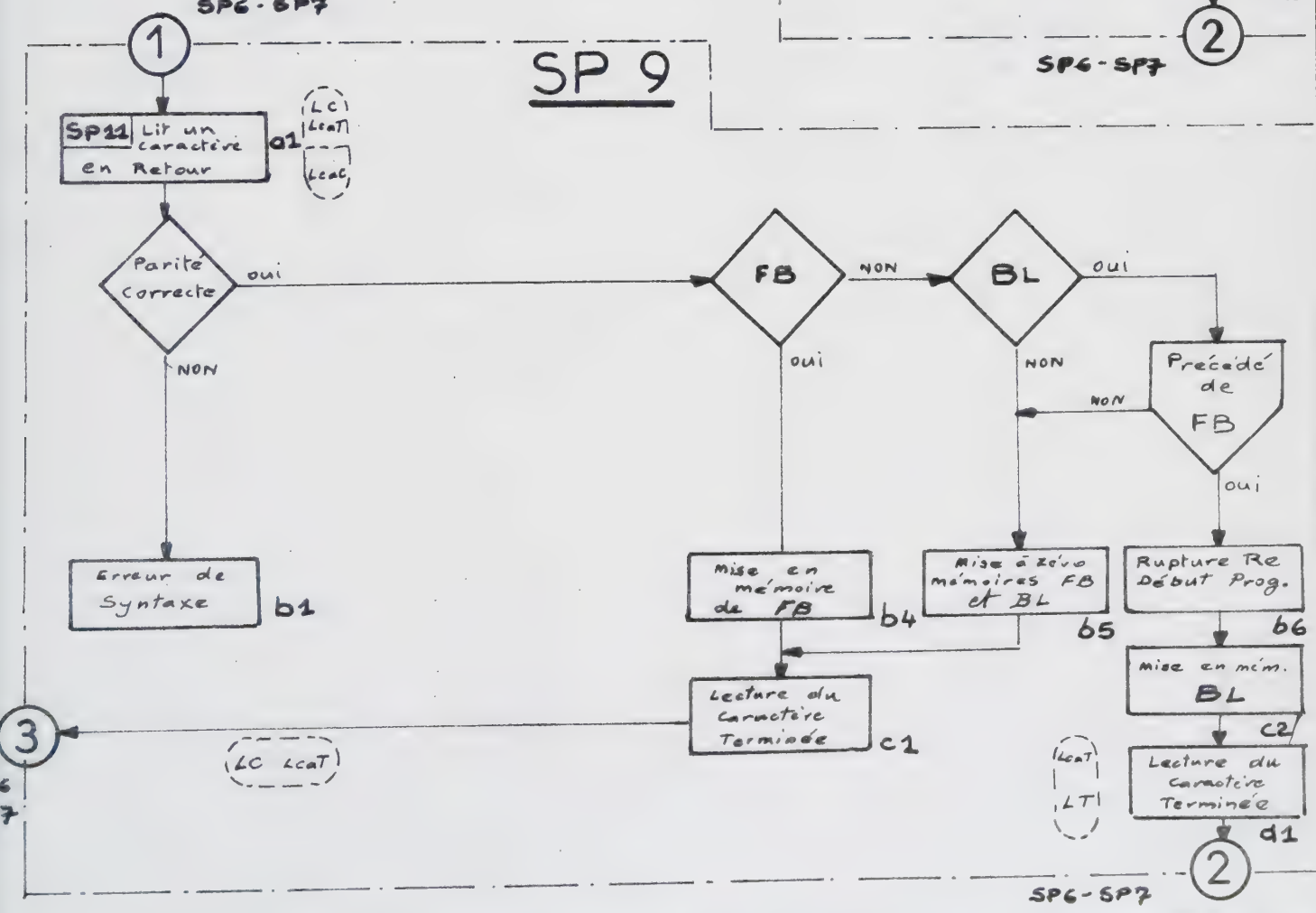
SP6-SP7

SP 8



SP6-SP7

SP 9



SP6-SP7

SP6-SP7

Sous-programmes des ruptures de début de programme, en mise en place de la bande

PLANCHE 5-7
Ordinogrammes
SP8-SP9-MO 1

SP2
SP5
SP8

1

SP10

Impulsion
Avance
Lecteur

a1

Echantillon.
Etat des
contacts

b1

Mise en mémoire
Tampon
du caractère

c1

Fin Echantillon.
Etat
des contacts

d1

2

SP2
SP5
SP8

SP4
SP9

1

SP11

Impulsion
Avance
Lecteur

a1

Echantillon.
Etat des
contacts

b1

Mise en mémoire
Tampon
du caractère

c1

Fin Echantillon.
Etat
des contacts

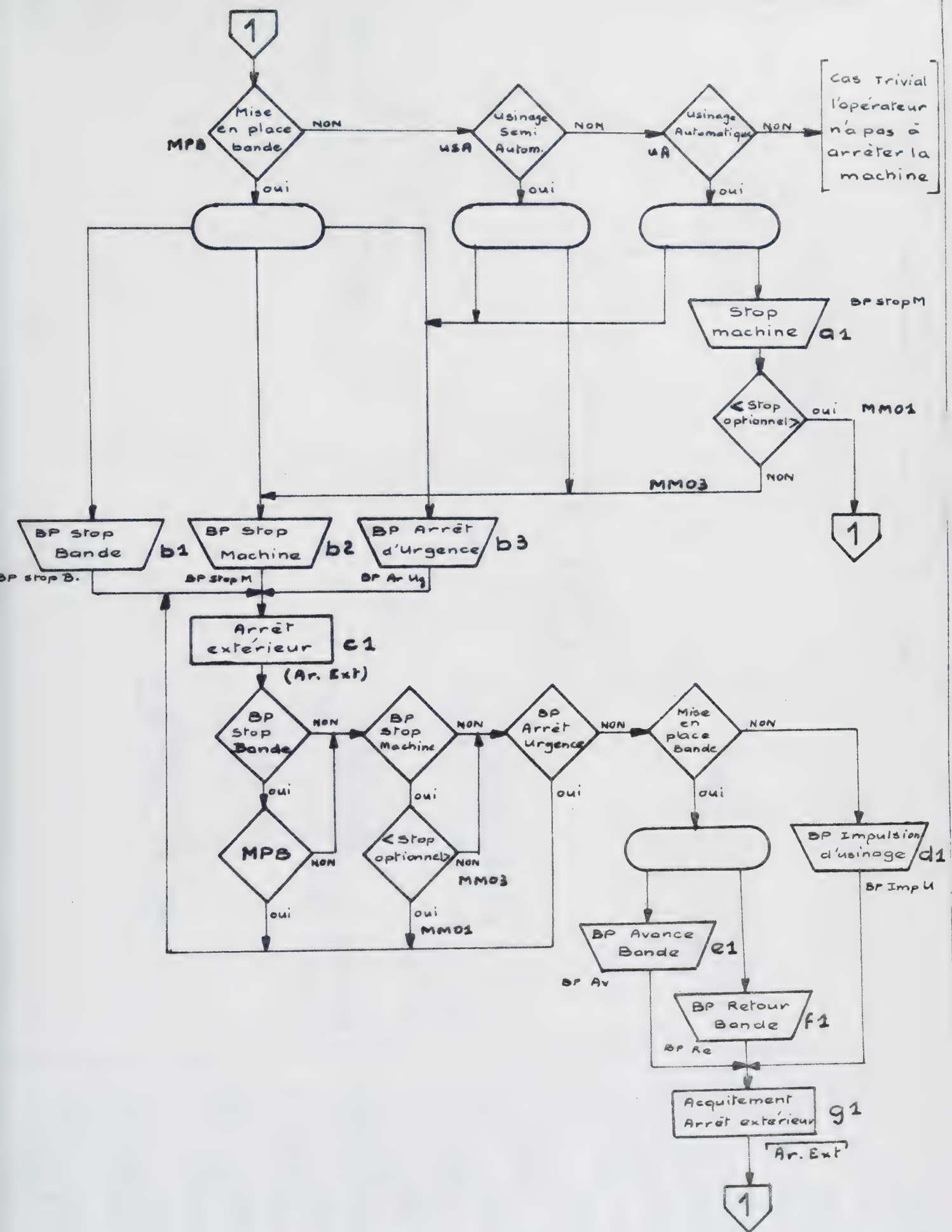
d1

2

SP4
SP9

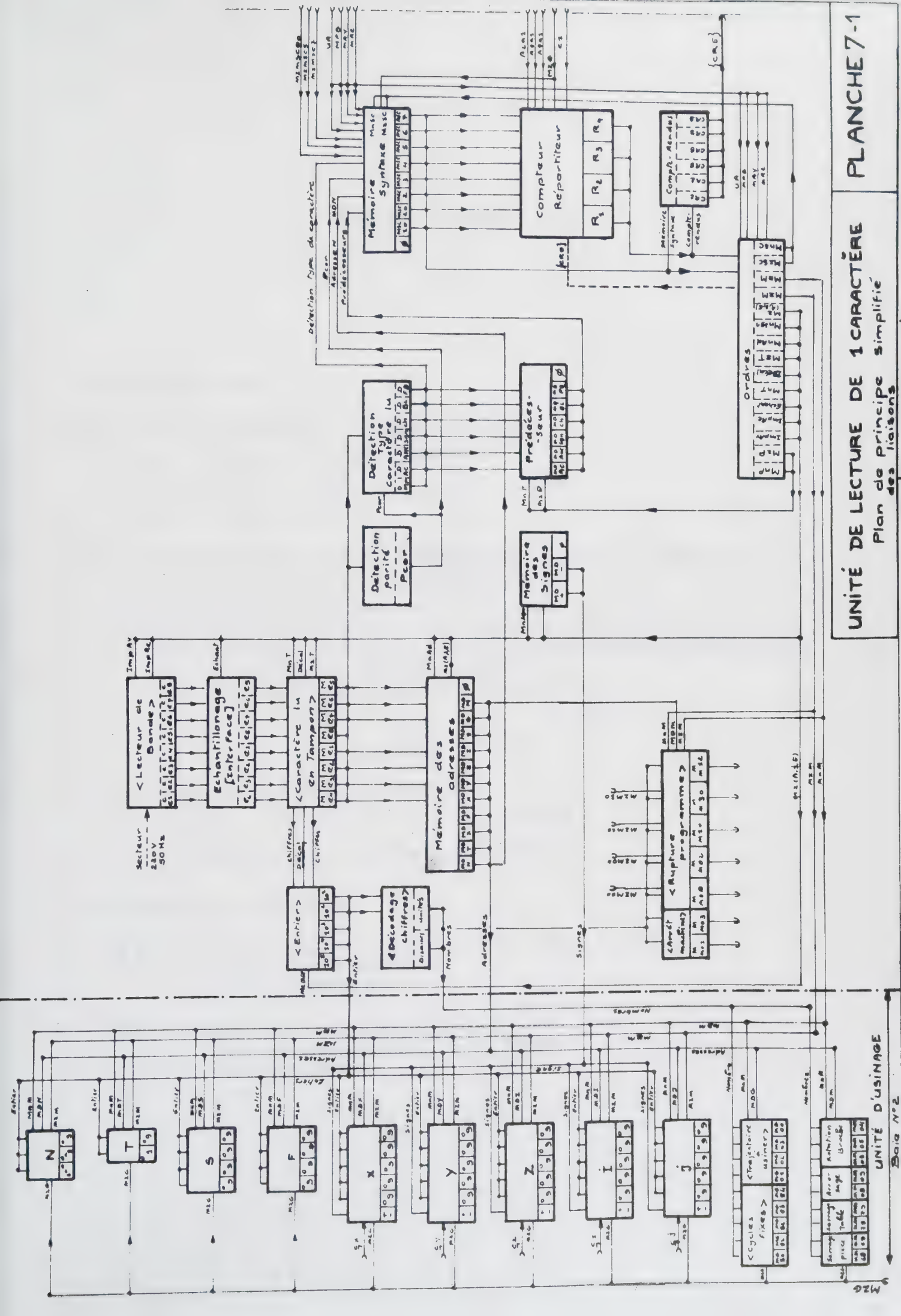
Sous-programmes de la lecture
de un caractère [en Avance
en Retour

PLANCHE 5-8
Ordinogrammes
SP10-SP11-MO1



Sous-programme des arrêts
et acquittement de ces arrêts

PLANCHE 5-9
Ordinogramme
SP 12-MO 1



UNITÉ DE LECTURE DE 1 CARACTÈRE
Plan de principe simplifié
Unité d'usage

ANNEXE TECHNOLOGIQUE

A-1- IntroductionA-2- Circuits logiques à niveau utilisés

- A-2-1- Portes à diodes (rappels)
- A-2-2- Régénérateurs
- A-2-3- Portes à diodes de base
- A-2-4- Conditionneurs

A-3- Types de "parasites" en fonction des modes de connection

- A-3-1- Rappels des types de parasite
- A-3-2- Caractéristiques des conducteurs utilisés
- A-3-3- Perturbations dans une ligne de transmission logique
- A-3-4- Perturbations entre lignes de transmission parallèles (parasites d'intermodulation).
- A-3-5- Perturbations dues à l'alimentation.

PLANCHES JOINTES

- A-1 Portes à diodes (rappels)
- A-2 Portes à diodes de base (utilisees)
- A-3 Conditionneurs à diodes (utilisees)

Bibliographie partielle :

- [1] Electronique des impulsions - Tome I et II
par Metzger et Vabre
Editeur Masson
- [2] Calcul des circuits à transistors
par les Ingénieurs de la Texas
Editeur Dunod.

A-1 INTRODUCTION

Nous avons signalé au début de cette thèse que les problèmes technologiques rencontrés lors de la conception d'une machine logique complexe sont nombreux et variés. Nous n'allons pas tous les aborder dans cette annexe.

1°) Nous présenterons les circuits logiques utilisés en nous limitant aux opérateurs à niveau. Il nous a semblé intéressant d'introduire la technique des conditionneurs à diode et surtout de "conditionneur type" réalisant une fonction logique de la forme :

$$D = \eta a. [\xi b + \eta c + \eta d \cdot \eta e + \eta f (\eta g + \eta h + \xi k)]$$

permettant d'obtenir des circuits logiques rapides comportant peu de régénérateurs et ayant un coût faible.

2°) Nous aborderons ensuite différents types de parasites dus aux modes de connection. Dans la plupart des cas pratiques, le calcul rigoureux des "parasites" est impossible, nous nous limiterons donc à des cas simples, calculés approximativement. Nous serons conduits à mettre ainsi en évidence une partie des règles empiriques respectées habituellement par les praticiens.

A-2 Circuits logiques à niveau utilisés

A-2-1- Portes à diodes (voir planche A-1)

- 1°) Porte à diodes POSITIVE : Une porte à diodes est dite POSITIVE si elle est conçue comme précisé ci-contre. L'on notera que la résistance de porte R_g est reliée à une source de potentiel POSITIVE : $+V_g$.
- 2°) Porte à diodes NEGATIVE : Une porte à diodes est dite NEGATIVE si elle est conçue comme indiqué sur le schéma ci-contre. La résistance de Porte R_g est connectée à une source de potentiel NEGATIVE : $-V_g$.
- 3°) Table des NIVEAUX : Si l'on applique à chaque entrée e_i , un signal booléen, de tension Ve_i , à deux valeurs possibles (ou Niveaux) notées B et H avec $-V_g \leq B < H \leq +V_g$, le potentiel V de la sortie S est donné pour chaque porte par la table aux Niveaux précisée sur la planche.

4°) Remarques :

。 L'on peut remplacer les potentiels $-V_g$ et $+V_g$ respectivement par les potentiels V_N et V_P tels que $V_N \leq B < H \leq V_P$ sans changer les tables aux niveaux des portes.

。 Les portes à diodes sont des circuits passifs, "dégénérant" les signaux. Par exemple, pour une porte positive dont une seule entrée (Ex. : e_1) est au niveau $Ve_1 = B$, les autres au Niveau H, nous avons :

$$V_S \approx Ve_1 + 0,7 + \frac{V_g - (0,7 + Ve_1)}{R_g + R_d} R_d$$

avec : $\left[\begin{array}{l} R_d : \text{Résistance directe de la diode} \approx 200 \Omega \\ 0,7V : \text{Seuil de la diode} \\ R_g : \text{Résistance de Porte} \approx 8,2 \text{ k}\Omega \end{array} \right.$

5°) Equation logique de ces portes :

A chaque signal logique de niveaux B et H, l'on peut associer un signal booléen de valeur 0 et 1.

- la logique est dite POSITIVE si à $\begin{cases} H & \text{l'on associe 1} \\ B & \text{l'on associe 0} \end{cases}$
- la logique est dite NEGATIVE si à $\begin{cases} H & \text{l'on associe 0} \\ B & \text{l'on associe 1} \end{cases}$

En fonction de l'association ci-dessus, le lecteur retrouvera aisément les équations des portes à diodes positives et négatives en fonction du choix fait pour l'association (logique positive, logique négative).

Dans le cas où une ou plusieurs entrées d'une porte ne sont pas utilisées l'on remarquera qu'il est possible de laisser ces entrées en l'air.

A-2-2- Régénérateurs (voir planche A-2)

Les régénérateurs sont les opérateurs classiques ci-contre. (Inverseurs et Amplificateurs) dont les niveaux des sorties S sont $B \approx 0V$ et $H \approx 8V$.

En fonction de la tension V_s appliquée à l'entrée s , nous avons si V_z est le potentiel zéner :

$$V_s \approx B \rightarrow V_{EB} \approx B - V_z \rightarrow \text{si } V_{EB} < 0 \quad \begin{array}{l} \text{transistor bloqué} \\ V_s \approx 8V \end{array}$$

$$V_s \approx H \rightarrow V_{EB} \approx H - V_z \rightarrow \text{si } V_{EB} > 0 \quad \begin{array}{l} \text{transistor passant.} \\ V_s \approx 0V \end{array}$$

Réjection de bruit : la tension V_z (si $B = 0\text{volts}$) est la tension dite de réjection du bruit du régénérateur.

Groupeement des régénérateurs : Leurs caractéristiques électroniques permettent de les connecter en parallèle (sorties court-circuitées)

Schéma logique				
Fonction en Logique	POSITIVE	$S = s_1 \cdot s_2$	$S = \overline{s_1} \cdot \overline{s_2}$	$S = \overline{s_1} \cdot s_2$
	NEGATIVE	$S = s_1 + s_2$	$S = \overline{s_1} + s_2$	$S = \overline{s_1} + \overline{s_2}$

A-2-3- Portes à diodes de base (voir planche A-2)

• Nous conviendrons d'étudier tous les circuits en logique positive $B \hat{=} 0$ logique ; $H \hat{=} 1$ logique

1°) Porte logique ET : C'est une porte POSITIVE classique

2°) Porte logique ET : C'est une porte POSITIVE dont la résistance R_g n'est pas câblée.

3°) Porte logique ou spéciale : C'est une porte NEGATIVE.

L'on remarquera sur la planche ci-contre que la résistance R_g est absente, la résistance R_b de polarisation de la base des régénérateurs devant obligatoirement faire office de résistance de porte négative.

4°) Porte logique OU : C'est une porte logique OU spéciale dont chaque entrée est connectée à la sortie d'une porte ET à une entrée.

Nota :

Toute entrée non utilisée d'un opérateur logique sera considérée comme à l'état 1 logique si elle est en l'air, et à l'état 0 logique si elle est connectée à la masse.

A-2-4- Conditionneurs (voir planche A-3)

Nous appelons conditionneur un ensemble de portes à diodes interconnectées.

Afin d'économiser des régénérateurs, de diminuer les temps de retard des signaux logiques (dus essentiellement aux régénérateurs), l'on recherche à utiliser des groupements en parallèle de portes à diodes.

Dans la planche ci-contre, nous avons représenté les groupements classiques dits conditionneurs de base, et le groupement type, de fonction logique maximum, que l'on peut obtenir.

Nous n'expliquerons pas les fonctionnements, le lecteur les retrouvera aisément à l'aide des schémas électroniques.

1°) Conditionneur de base :

Conditionneur ET/ET : Extension du nombre des entrées d'un ET par des opérateurs ET. (L'extension d'un ET par un ET est possible, bien que déconseillée, la résistance équivalente de porte étant alors $R_g/2$).

Conditionneur OU/OU : Extension d'un OU par des opérateurs OU.

Conditionneur OU/ET_Δ : Extension d'un OU par des opérateurs ET_Δ. L'on remarquera que si une entrée du ET_Δ est au niveau $B \hat{=} 0$ logique, le potentiel de S est B, quels que soient les niveaux des entrées du OU.

En S (sortie du conditionneur) deux extensions sont possibles :

- Extension du OU par des opérateurs OU, OU Spécial, ET câblé 1/2 ou Fantôme.

- Extension du ET_Δ par des opérateurs ET_Δ.

Conditionneur ET/OU Spécial : L'on obtient un conditionneur analogue à la porte logique OU, sauf que les opérateurs ET ont plusieurs entrées au lieu d'une seule.

Conditionneur ET câblé 1/2 ou Fantôme : La sortie S est la branche de connection d'entrée d'opérateurs ET. (Le schéma électronique est identique aux conditionneurs ET/OU Spécial).

Pour les conditionneurs ET/OU Spécial et ET câblé 1/2 OU Fantôme, l'on remarquera que :

- En S l'on peut effectuer les extensions classiques d'un, ou avec :

- les opérateurs OU
- les opérateurs ET_Δ
- les opérateurs ET câblés 1/2 ou Fantôme.

- En Sa ou Sb l'on peut effectuer l'extension classique avec les opérateurs ET_{Δ} .

2°) Conditionneur type (voir planche A-3)

Nous avons représenté le conditionneur type que l'on peut réaliser à l'aide des portes à diodes, dites de base. Il est obtenu par extension et association des conditionneurs de base ci-dessus.

Trois types d'extension essentiels de la sortie du conditionneur sont possibles :

- Extension par opérateur ET_{Δ} type πa
- Extension par opérateur OU type Σb
par opérateur OU Spécial type $(\pi c + \pi d)$
par opérateur ET câblé 1/2 ou Fantôme (identique à l'opérateur ou Spécial).
- Extension ET_{Δ} câblé en 1/2 ou Fantôme, type $\pi f \& k, \dots$

Cette extension introduit une remontée du niveau bas (diode supplémentaire) d'où une diminution du Niveau de Réjection de bruit.

L'équation type des conditionneurs à diode est :

- En logique POSITIVE ($B \hat{=} 0$ logique ; $H \hat{=} 1$ logique)

$$S = \pi a \cdot [\Sigma b + \pi c + \pi d \cdot \pi e + \pi f (\pi g + \pi h + \Sigma k)]$$

- En logique NEGATIVE ($B \hat{=} 1$ logique ; $H \hat{=} 0$ logique)

$$S = \Sigma a + \pi b \cdot \Sigma c \cdot (\Sigma d + \Sigma e) \cdot [\Sigma f + \Sigma g \cdot \Sigma h \cdot \pi k]$$

La connection de ce conditionneur à diodes aux différents régénérateurs, ou de plusieurs conditionneurs à un groupement de régénérateurs, permet de réaliser les fonctions logiques combinatoires.

A-3 Types de parasites en fonction du mode de connection

A-3-1- Rappels des types de parasites :

Dans un système logique, comportant un nombre élevé d'opérateurs, répartis dans plusieurs armoires, il est primordial lors de la détermination des dossiers de câblage de tenir compte de règles plus ou moins empiriques afin de limiter les fonctionnements anormaux dus aux parasites venant perturber les informations logiques.

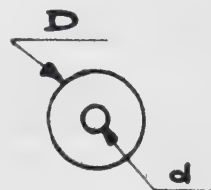
Nous allons étudier, par des calculs très simplifiés, les types de perturbations que l'on rencontre couramment, c'est à dire :

- les perturbations dans une ligne de transmission (propagation d'une onde dans une ligne).
- les perturbations entre lignes de transmission parallèles dénommées usuellement parasites d'intermodulation, dus aux couplages électrostatiques et électromagnétiques entre lignes.
- les perturbations dues aux réseaux d'alimentation logique.

A-3-2- Caractéristiques des conducteurs utilisés :

1°) Câbles coaxiaux :

[Diélectrique : TFE (Polytétrafluoréthylène)
 $\epsilon_r = 2$ F/m
 $d = 0,5\text{mm}$; $D = 1,5\text{mm}$



$$L = 4,6 \cdot 10^{-7} \cdot \log \frac{D}{d} \quad \text{H/m} \qquad L = 22 \cdot 10^{-7} \quad \text{H/m}$$

$$C = 24 \cdot 10^{-12} \cdot \epsilon_r / \log \frac{D}{d} \quad \text{F/m} \qquad C = 100 \quad \text{F/m}$$

$$R_c = \frac{138}{\sqrt{\epsilon_r}} \cdot 10^{-3} \cdot \log \frac{D}{d} \quad \text{/m} \qquad R_c = 50 \quad \Omega/\text{m}$$

$$v = \frac{C}{\sqrt{\epsilon_r}} \qquad v = 0,212 \quad \text{m/ns}$$

$$R = \rho \frac{1}{s} \qquad R = 80 \cdot 10^{-3} \quad \Omega/\text{m}$$

2°) Ligne simple parallèle au plan de masse :

$$d = 0,75 \text{ mm} \qquad h = 5\text{mm}$$

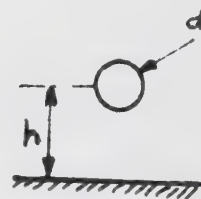
$$L = 4,6 \cdot 10^{-7} \log \frac{4h}{d} \qquad L = 6,56 \cdot 10^{-7} \quad \text{H/m}$$

$$C = 127 \cdot 10^{-12} / \log \frac{4h}{d} \qquad C = 90 \cdot 10^{-12} \quad \text{F/m}$$

$$R_c = 138 \cdot \log \frac{4h}{d} \qquad R_c = 197 \quad \Omega/\text{m}$$

$$v = 1/\sqrt{LC} \qquad v = 0,132 \quad \text{m/ns}$$

$$R = \rho \frac{1}{s} \qquad R = 53 \cdot 10^{-3} \quad \Omega/\text{m}$$



3°) Ligne bifilaire symétrique :

$$d = 0,75 \text{ mm}$$

$$D = 3 \text{ mm}$$

$$L = 9,2 \cdot 10^{-7} \log \frac{2D}{d}$$

$$L = 8,3 \cdot 10^{-7} \text{ H/m}$$

$$C = 63 \cdot 10^{-12} / \log \frac{2D}{d}$$

$$C = 70 \cdot 10^{-12} \text{ F/m}$$

$$R_c = 138 \log \frac{2D}{d}$$

$$R_c = 250 \text{ } \Omega/\text{m}$$

$$v = 1/\sqrt{LC}$$

$$v = 0,13 \text{ m/ns}$$

$$R = \rho \frac{e}{s}$$

$$R = 106 \cdot 10^{-3} \text{ } \Omega/\text{m}$$

4°) Remarques :

a) Nous venons de rappeler les caractéristiques de différents conducteurs utilisés en supposant que la répartition du courant dans ces conducteurs était uniforme (basse fréquence). Dans le cas de la propagation d'un train d'impulsion logique, nous avons des transitions de courtes durées. La répartition du courant va évoluer de la périphérie vers une répartition uniforme.

A toutes les caractéristiques ci-dessus, il conviendra d'ajouter des termes correctifs dus :

- à l'effet pelliculaire (épaisseur de la peau)
- aux effets de proximités (modifiant la répartition du courant dans un conducteur en fonction des conducteurs voisins).

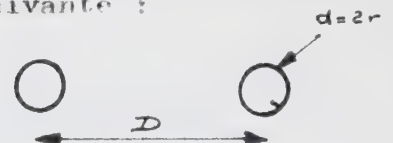
b) Les caractéristiques des lignes sont fonction de la fréquence. En étude impulsionnelle les modèles classiques du régime sinusoïdal ne sont pas valables.

En régime impulsionnel, l'on détermine une épaisseur de peau opérationnelle $\delta = \sqrt{\frac{2}{\mu \sigma f}}$ d'où une modification de la constante de temps de propagation :

$$\gamma = \sqrt{(R_a + L_p) C_p} \quad \text{avec } R_a = K \sqrt{p}$$

$$\gamma = p \sqrt{L_c} \cdot \sqrt{1 + \frac{K}{L \sqrt{p}}}$$

c) l'on peut tenir compte de la modification des valeurs de l'inductance et de la capacité linéique due aux effets pelliculaires, par exemple pour la ligne suivante :



Self inductance basse fréquence

$$L = 9,2 \cdot 10^{-7} \log \frac{2D'}{d} \quad \text{avec } D' = \frac{D}{2} \left(1 + 1 - \left(\frac{d}{D} \right)^2 \right)$$

Self inductance due à l'effet de peau :

$$L_i = \frac{10^{-4}}{d \sqrt{1 - \left(\frac{d}{D} \right)^2}} \times \sqrt{\frac{\mu_r}{\sigma f}} \quad \begin{array}{l} \mu_r = 1 \text{ dans l'air} \\ f = \text{fréquence} \\ \sigma : \text{conductivité} \end{array}$$

d'où la self inductance à la fréquence f :

$$L_T = L + 2L_i$$

A-3-3- Perturbations dans une ligne de transmission :

Considérons un régénérateur (transistor) relié à une porte logique par l'intermédiaire d'un fil de liaison L_1 . L_1 se comporte comme une ligne, en général non adaptée. Les transitions (enclenchements, et déclenchements) du régénérateur impliquent un régime transitoire dans la ligne pouvant entraîner des fonctionnements anormaux de la porte et même du régénérateur si celui-ci est une bascule.

Nous allons étudier succinctement le montage suivant comportant un régénérateur relié à n portes groupées, par l'intermédiaire de la ligne L_1 , d'origine E et d'extrémité R .



Pour un régénérateur et une porte les capacités parasites de connexion (C_c) sont sensiblement égales à 20 pF.

1°) fonctionnement en régime permanent :

a) au niveau Haut : H, le transistor est bloqué.
Pour m portes à l'état $s = H$ nous avons :

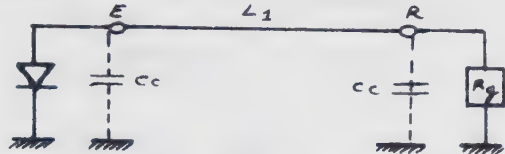
$$\left[\begin{aligned} V_H &\approx V_g + (V_T - V_g) \frac{R_g/m}{R_T + R_g/m} \\ i_H &\approx \frac{V_T - V_g}{R_T + R_g/m} \end{aligned} \right.$$

b) au niveau Bas : B, le transistor conduit. Pour n portes reliées au régénérateur, nous avons :

$$\left[\begin{aligned} V_B &\approx 0 \text{ volts} \\ i_B &\approx \frac{V_g}{R_g/n} \quad \text{soit } n \text{ fois le courant de 1 charge.} \end{aligned} \right.$$

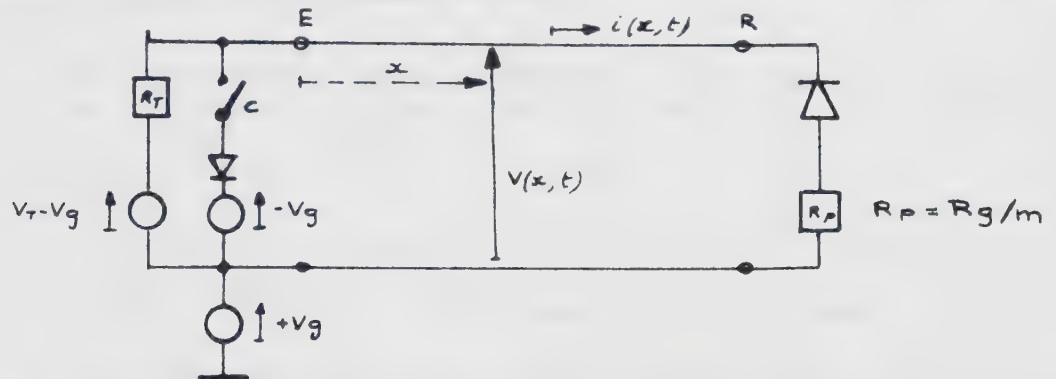
c) remarques :

En fonction du niveau H ou B de la ligne L_1 , le montage se comporte pour les perturbations extérieures comme il est précisé sur les schémas suivants :

au niveau B	au niveau H
 Ce schéma est valable tant que $V_A \text{ parasite} < V_g$	 Ce schéma est valable tant que $V_R \text{ parasite} < (+V_g - V_H)$

2°) Fonctionnement en commutation :

Après avoir supposé le transistor parfait, négligé les capacités parasites et effectué une translation des potentiels de $-V_g$, nous sommes conduits au schéma suivant :



Nous allons appliquer la méthode des ondes mobiles afin d'étudier les effets des transitions de l'émetteur (fermeture et ouverture du contact c), en supposant que les pertes en ligne et les variations des constantes linéiques dues aux effets pelliculaires sont négligeables.

a) Caractéristiques générales de fonctionnement :

Les régimes permanent de la ligne sont :

Conditions initiales C fermé conditions finales C ouvert	Conditions initiales C ouvert conditions finales C fermé
$V_E = V_R = (V_T - V_g) \frac{R_p}{R_T + R_p}$ $i_E = i_R = \frac{V_T - V_g}{R_p}$	$V_E = V_R = -V_g$ $i_E = i_R = -\frac{V_g}{R_p}$

Les coefficients de réflexion Γ_E et Γ_R à l'émetteur E et au récepteur R, respectivement sont :

Emetteur		Récepteur
Contact C ouvert	Contact C fermé	
$\Gamma_E = \frac{R_T - R_C}{R_T + R_C}$	Si $V_E < -V_g$ $R_E = \infty \rightarrow \Gamma_E = +1$	Si $V_R < 0$ $\Gamma_R = \frac{R_p - R_c}{R_p + R_c}$ $\Gamma_R > 0$ si $R_p > R_c$
$\Gamma_E > 0$ si $R_T > R_C$ $\Gamma_E = 0$ si $R_T = R_C$ $\Gamma_E < 0$ si $R_T < R_C$	Si $V_E > -V_g$ $R_E = 0 \rightarrow \Gamma_E = -1$	Si $V_R > 0$ $R_p = \infty \rightarrow \Gamma_R = +1$

En général la ligne logique L_1 n'est pas adaptée, dans les cas de conduction des diodes R_T et R_p sont tels que :

$$R_T < R_C \rightarrow \Gamma_E > 0 \quad R_p > R_c \rightarrow \Gamma_R > 0$$

La durée de propagation d'une onde dans la ligne est :

$$\tau = \frac{\ell}{v} = \ell \sqrt{LC} \quad (\ell : \text{longueur de la ligne})$$

b) Etudions les effets d'un déclenchement (transition : H $\rightarrow$ B, correspondant à la fermeture du contact C).

Du début de la perturbation (époque 0) à l'époque 2τ , seule l'onde incidente v_i agit sur l'entrée E, nous avons :

$$V_E = -V_g = v_E^0 + v_i \quad \text{avec } v_E^0 = (V_T - V_g) \frac{R_p}{R_p + R_T}$$

$$v_i = -V_g - (V_T - V_g) \frac{R_p}{R_p + R_T} = -V_H$$

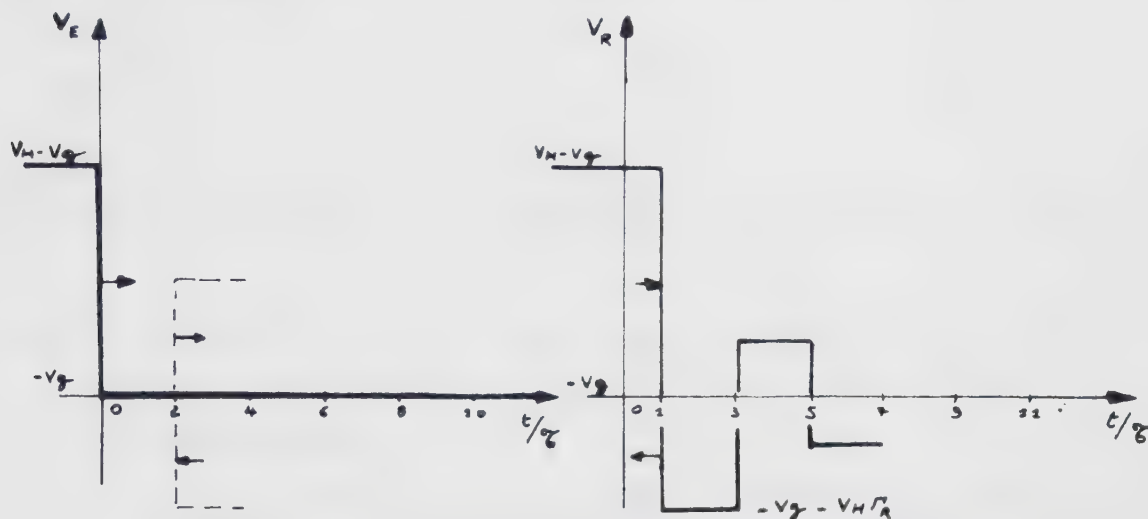
Ensuite nous avons les évolutions de la tension en E et R avec :

$$\Gamma_E = -1 \text{ et } \Gamma_R = \frac{R_p - R_c}{R_p + R_c} > 0 \text{ dans les cas de conduction des diodes.}$$

	Emetteur E		Récepteur R	
	Tension V_E	Ondes	Ondes	Tension V_R
0 ⁻ 0 ⁺ τ	$V_H - V_g$ $-V_g$	$-V_H$	$-V_H$ $-V_H \cdot \Gamma_R$	$V_H - V_g$ $-V_g - V_H \cdot \Gamma_R$
2τ	$-V_g$	$-V_H \cdot \Gamma_R$ $+V_H \cdot \Gamma_R$	$+V_H \cdot \Gamma_R$ $+V_H \cdot \Gamma_R^2$	$-V_g + V_H \cdot \Gamma_R^2$
3τ				
$\infty \tau$	$-V_g$			$-V_g$

La valeur limite de l'onde est $-V_H \times \frac{1 + \Gamma_R}{1 - \Gamma_E \cdot \Gamma_R} = -V_H$

Les signaux V_E et V_R se représentent en fonction du temps réduit $t_R = \frac{t}{\tau}$ de la façon suivante (avec $\Gamma_R = 1/2$).



c) Etudions les effets d'un enclenchement (transition B $\rightarrow$ H, correspondant à l'ouverture du contact C).

Du début de la perturbation (époque 0^+) à l'époque 2, nous avons à l'entrée E :

$$I_E = \frac{(V_T - V_g) - V_E}{R_T} = \frac{V_E - V_E^0}{R_C} + i_E^0 \quad V_E = V_E^0 + V_1 + V_r$$

$$\text{Soit : } v_1 = \frac{R_C}{R_C + R_T} \left[(V_T - V_g) + V_g + \frac{R_T \cdot V_g}{R} \right]$$

$$v_1 = \frac{R_C}{R_C + R_T} \left[(V_T - V_g) + V_g \frac{R_P + R_T}{R_P} \right]$$

D'où la valeur limite de l'onde :

$$\begin{aligned} v_1 \times \frac{1 + \Gamma_R}{1 - \Gamma_E \cdot \Gamma_R} &= \frac{R_P}{R_T + R_P} \times \frac{R_C + R_T}{R_C} v_1 \\ &= (V_T - V_g) \frac{R_P}{R_T + R_P} + V_g \end{aligned}$$

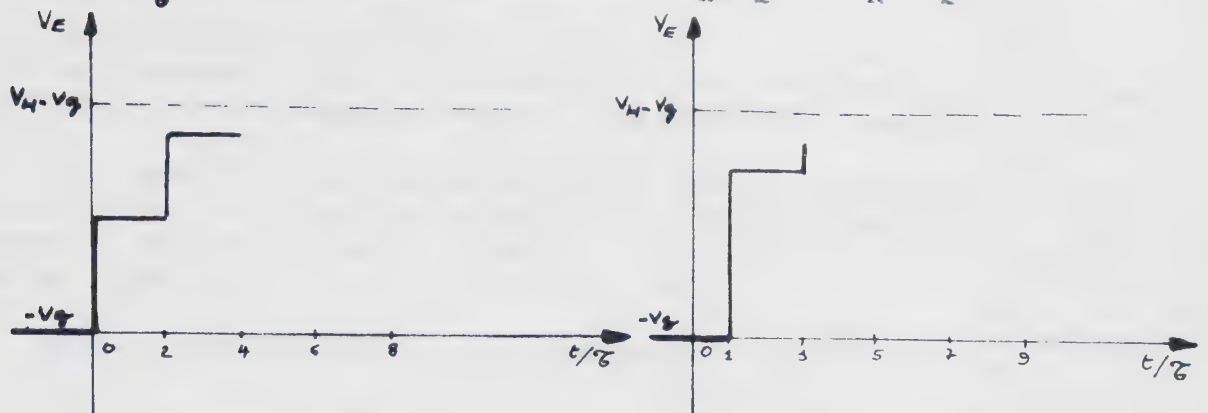
La tension finale est donc $V_E = V_R = -V_g + (V_T - V) \frac{R_P}{R_T + R_P} + V_g$

Après avoir établi le tableau d'évolution des ondes avec

$$\Gamma_E = \frac{R_T - R_C}{R_T + R_C} > 0 \quad \text{et} \quad \Gamma_R = \frac{R - R_C}{R + R_C} > 0 \quad \text{l'on peut représen-}$$

ter les tensions V_E et V_R en fonction du temps réduit :

$t_R = \frac{t}{\tau}$ de la façon suivante ($\Gamma_E = \frac{1}{2}$; $\Gamma_R = \frac{1}{2}$)



$$\text{avec : } V_E = \frac{R_C}{R_C + R_T} (V_T - V_g) + V_g \times \frac{R_T}{(R_T + R_C)} \times \frac{R_C - R_P}{R_P}$$

d) Conclusion :

1°) Dans le cas des lignes longues (la durée de propagation τ est plus grande que le temps de montée ou de descente, à la tension de réjection du régénérateur piloté), les signaux $V_H \cdot \Gamma_R^2$ peuvent introduire des transitions imprévues si leur niveau est supérieur au niveau de réjection du récepteur.

Il est nécessaire de diminuer la résistance R_P afin d'approcher de l'adaptation de la ligne $R_P \approx R_C^P$ par adjonction en fin de ligne d'une résistance polarisée à $+V_g$ (l'on remarquera que la sortance de l'émetteur est diminuée). L'adaptation étant impossible pour toutes les lignes, l'on introduit à chaque porte d'entrée une diode polarisée à zéro volt telle que les surtensions négatives soient "écrêtées". L'adaptation de la ligne, à l'émetteur est aussi nécessaire (réseau RL, diode de Clamp).

2°) Dans le cas des lignes courtes (le double de la durée de propagation est faible par rapport au temps de montée ou de descente), l'énergie de l'impulsion $V_H \cdot \Gamma_R^2$ est insuffisante pour changer l'état du régénérateur piloté (charge des capacités).

3°) L'on remarquera que pour la logique considérée les flancs de montée sur circuits désadaptés sont dégénérés (par la ligne parfaite, par ses effets pelliculaires). L'on a intérêt à transmettre des signaux actifs par leurs flancs de descente.

A-3-4- Perturbations entre lignes de transmissions parallèles : (parasites d'intermodulation).

Lorsque deux lignes L_1 et L_2 de transmission d'informations logiques passent l'une à côté de l'autre, nous avons un couplage magnétique et un couplage électrostatique entre ces deux lignes.

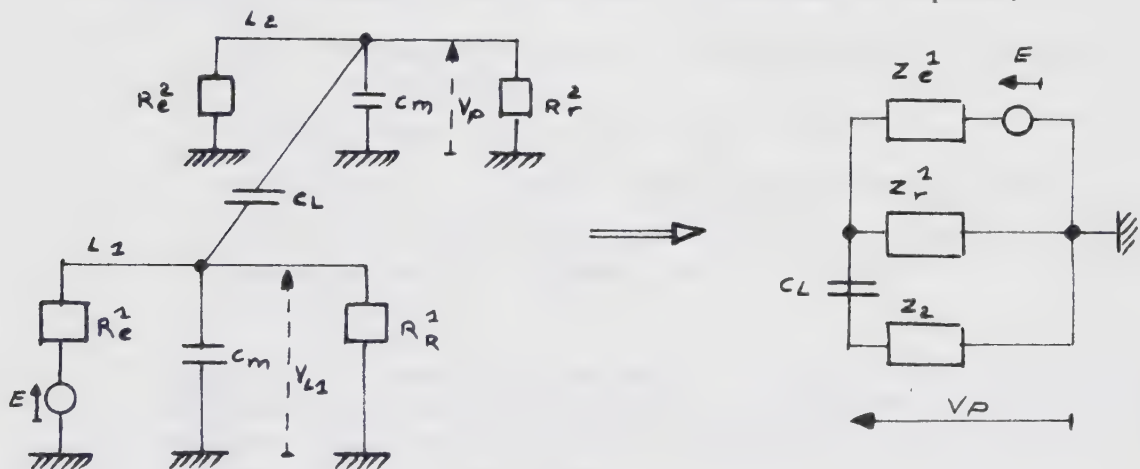
La ligne L_1 étant excitée par un train d'impulsions logique, nous allons rechercher la valeur de la tension parasite V_p qui apparaît aux bornes de la ligne L_2 en fonction des différents types de couplage.

Nous allons étudier successivement :

- les couplages basse fréquence (1er harmonique du train d'impulsions)
- le couplage haute fréquence (transmission d'un flanc de montée ou de descente).

A-3-4-1- Couplage basse fréquence :

1°) Couplage capacitif : L'ensemble des capacités entre les lignes L_1 L_2 peut être ramené à la capacité C_L dite capacité de couplage, figurée sur le schéma ci-après.



La ligne L_1 est excitée par une tension sinusoïdale E de basse fréquence (1er harmonique d'un train d'impulsions) nous allons rechercher la valeur de la tension parasite V_p qui apparaît aux bornes de la ligne L_2 .

$$\text{Soit : } V_p = E \times \frac{Z_r^1 \cdot Z_2}{Z_e^1 \cdot Z_r^1 + (Z_e^1 + Z_r^1) \left(Z_2 + \frac{1}{C_p} \right)}$$

avec :

$$\frac{1}{Z_2} = \frac{1}{R_2^2} + \frac{1}{R_2^2} + C_{mp}$$

En fonction du potentiel V_{L1} de la ligne L_1 , V_p s'écrit :

$$V_p = V_{L1} \times \frac{Z_2}{Z_2 + \frac{1}{C_L p}}$$

$$\text{D'où : } V_p = V_{L1} \times \frac{A \cdot C_L \cdot p}{B \cdot C_L \cdot p + 1}$$

$$\text{avec } A = \frac{R_{\bullet}^2 \cdot R_r^2}{R_{\bullet}^2 + R_r^2} ; B = A \left(1 + \frac{C_m}{C_L} \right)$$

$$\text{Soit en module : } \left| \frac{V_p}{V_{L1}} \right| = \frac{A \cdot C_L \cdot \omega}{\sqrt{(B \cdot C_L \cdot \omega)^2 + 1}} \quad p = j\omega$$

$$\left| \frac{V_p}{V_{L1}} \right| = \frac{A \cdot C_L \cdot \omega}{\sqrt{\left[\left(1 + \frac{C_m}{C_L} \right) A \cdot C_L \cdot \omega \right]^2 + 1}}$$

Pour les basses fréquences c'est à dire pour tel que :

$$\left[A \cdot C_L \cdot \left(1 + \frac{C_m}{C_L} \right) \right]^2 \ll 1, \text{ nous avons :}$$

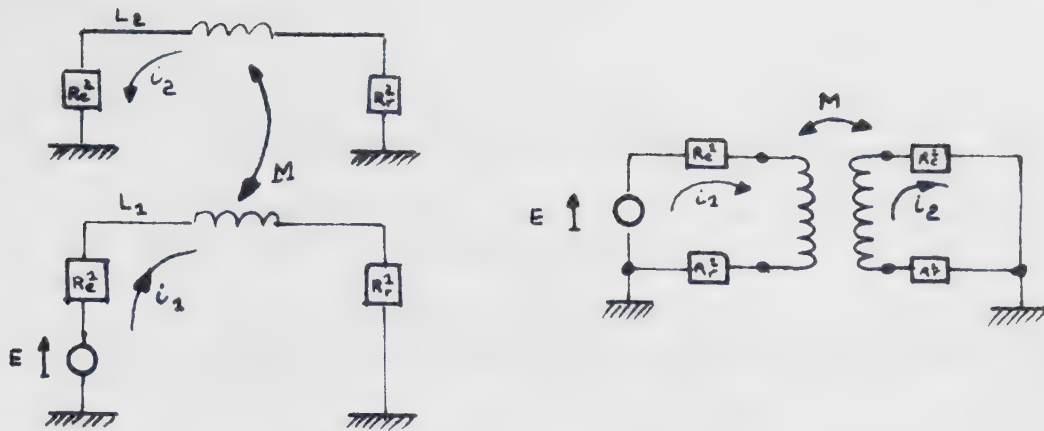
$$\left| \frac{V_p}{V_{L1}} \right| \approx A C_L \omega$$

Si l'on souhaite diminuer l'amplitude de la tension parasite V_p , l'on peut soit :

- Diminuer les résistances $R_{\bullet}^2$ et R_r^2
- Diminuer à la rigueur une résistance $R_{\bullet}^2$ ou R_r^2
- Diminuer la capacité parasite C_L en augmentant la distance entre les lignes, ou en interposant un écran électrostatique.

Nota : Les capacités C_m entre les lignes et la masse comportent la capacité propre à la ligne et les capacités de câblage (les capacités des opérateurs par rapport à la masse, et les capacités des connecteurs sont de l'ordre de 10 à 20 pF).

2*) Couplage magnétique : L'ensemble des mutuelles inductions entre les lignes L_1 et L_2 peut être localisé sur les lignes L_1 et L_2 .



La ligne L_1 étant excitée par une tension sinusoïdale E de basse fréquence, nous allons calculer les valeurs des tensions parasites V_p qui apparaissent aux extrémités de la ligne L_2 . Nous négligerons volontairement les inductances des lignes.

$$i_2 = \frac{E - jM \cdot \omega}{(R_e^1 + R_r^1)(R_e^2 + R_r^2) + M^2 \omega^2}$$

$$V_R^2 = \frac{R_r^2 \cdot E \cdot jM}{(R_e^1 + R_r^1)(R_e^2 + R_r^2) + M^2 \omega^2}$$

$$\text{Soit : } V_R^2 = -I_n \times \frac{R_r^2}{R_e^2 + R_r^2} \times \frac{jM\omega}{1 + \frac{M^2 \omega^2}{(R_e^1 + R_r^1)(R_e^2 + R_r^2)}}$$

$$\text{avec } I_n = \frac{E}{R_e^1 + R_r^1} \quad (\text{courant dans la ligne } L_1 \text{ seule})$$

Pour les basses fréquences, c'est à dire si :

$$\frac{M^2 \omega^2}{(R_e^1 + R_r^1)(R_e^2 + R_r^2)} \ll 1 \quad \text{nous avons :}$$

$$V_R^2 = -I_n \times \frac{R_r^2}{R_e^2 + R_r^2} \times jM\omega$$

Si l'on souhaite diminuer la tension parasite due au couplage électromagnétique en basse fréquence l'on a intérêt à rendre le rapport $\frac{R_e^2}{R_r^2}$ maximum, c'est à dire :

- augmenter la résistance de l'émetteur, R_e^e
- diminuer la résistance du récepteur, R_r^e

L'on notera que ce résultat classique est en accord avec la réduction du couplage capacitif.

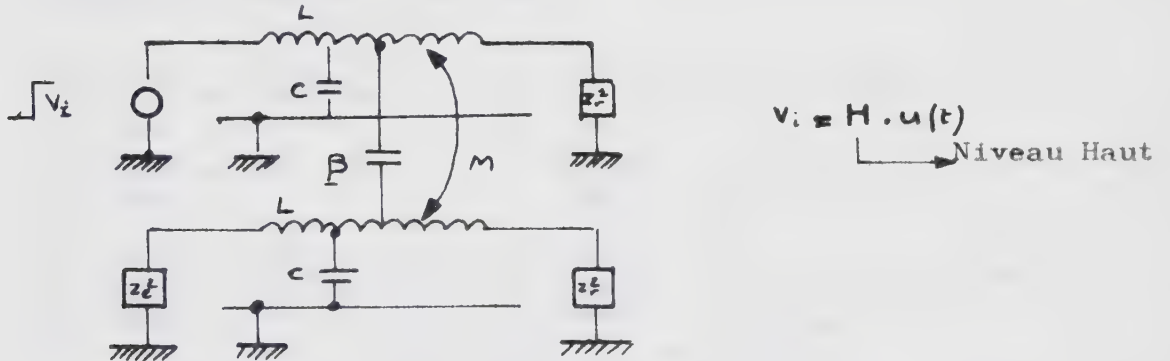
L'on peut diminuer la tension parasite de couplage magnétique en basse fréquence, en diminuant la mutuelle inductance c'est à dire :

- en écartant les lignes
- en interposant un écran électromagnétique
- en utilisant des fils torsadés permutés en leur milieu.

A-3-4-2- Couplage Haute Fréquence :

Une transition (enclenchement ou déclenchement), d'une information logique dans la ligne L_1 , perturbe la ligne L_2 . Nous allons pour une transition 0 → 1 (enclenchement) déterminer l'amplitude du parasite, et en rechercher sa forme afin de pouvoir déterminer les effets sur le récepteur de la ligne L_2 perturbée.

Soit deux lignes L_1 et L_2 , la ligne L_2 étant à un niveau logique déterminé, la ligne L_1 étant soumise à un échelon de tension.



Afin de simplifier la mise en équation du problème, nous allons supposer que :

- les lignes L_1 et L_2 sont identiques :
même longueur
même capacité et inductance linéique (C et L)
- portes en lignes nulles
- les lignes sont adaptées aux extrémités sur $R_c = \sqrt{\frac{L}{C}}$

- les couplages sont faibles ; les constantes linéiques sont : β : capacité ; M : mutuelle inductance.
- nous négligerons la variation des constantes linéiques en fonction de la fréquence.

Pour les charges électrostatiques nous pouvons écrire pour l'unité de longueur, à une époque t, à la distance x :

$$\frac{\partial q_j}{\partial t} = \frac{\partial I_j}{\partial x} \quad j = (1,2) \quad \text{ligne } L_1, \text{ ligne } L_2$$

$$\begin{aligned} Q_1 &= C_{11} V_1 + C_{12} V_2 \\ Q_2 &= C_{21} V_1 + C_{22} V_2 \end{aligned} \quad \text{avec} \quad \begin{cases} C_{11} = C_{22} = C \\ C_{12} = C_{21} = \beta \end{cases}$$

Pour les charges magnétiques nous avons :

$$\frac{\partial A_j}{\partial t} = - \frac{\partial V_j}{\partial x} \quad j = (1,2)$$

$$\begin{aligned} A_1 &= L_{11} I_1 + L_{12} I_2 \\ A_2 &= L_{21} I_1 + L_{22} I_2 \end{aligned} \quad \text{avec} \quad \begin{cases} L_{11} = L_{12} = L \\ L_{21} = L_{12} = M \end{cases}$$

Les couplages étant supposés faibles $\beta \ll C$ et $M \ll L$ après avoir négligé le couplage $L_2 \rightarrow L_1$, nous aurons pour échelon de tension d'entrée :

$$\begin{aligned} \left[\begin{aligned} - \frac{\partial I_1}{\partial x} &= C \rho V_1 \\ - \frac{\partial V_1}{\partial x} &= L \rho I_1 \\ \frac{\partial^2 V_1}{\partial x^2} - \frac{\rho^2}{v^2} V_1 &= 0 \end{aligned} \right] & \quad \left[\begin{aligned} - \frac{\partial I_2}{\partial x} &= C \rho V_2 - \beta \rho V_1 \\ - \frac{\partial V_2}{\partial x} &= L \rho I_2 + M \rho I_1 \\ \frac{\partial^2 V_2}{\partial x^2} - \frac{\rho^2}{v^2} V_2 &= \frac{\rho^2}{v^2} \times \frac{\beta}{C} (A-1) V_1 \end{aligned} \right] \\ A &= \frac{M}{L} \times \frac{C}{\beta} & v &= \sqrt{\frac{1}{LC}} \end{aligned}$$

L'équation de la propagation des ondes dans la ligne L_1 est la solution de $\frac{\partial^2 V_1}{\partial x^2} - \frac{\rho^2}{v^2} V_1 = 0$ $V_1 = H \cdot e^{-\rho \frac{x}{v}}$!
L étant la longueur de la ligne $\frac{L}{v} = T$ (durée de propagation) nous pouvons écrire : $V_1^1 = H_0 \cdot U(t-T)$

L'équation de la propagation des ondes dans la ligne L_2 est la solution de :

$$\left[\begin{aligned} \frac{\partial^2 V_2}{\partial x^2} - \frac{\rho^2}{v^2} V_2 &= \frac{\rho^2}{v^2} \times \frac{\beta}{C} \times (A-1) V_1 \\ V_1 &= H \cdot e^{-\rho \frac{x}{v}} \end{aligned} \right]$$

$$\text{Soit : } V_2 = \frac{\beta}{c} \times \frac{H}{2} \times \frac{A+1}{2} \left[e^{-\rho \frac{x}{v}} - e^{-\rho \frac{2L-x}{v}} \right] - \frac{\beta}{c} \times \frac{x}{v} \times \frac{A-1}{2} \cdot \rho \cdot H \cdot e^{-\rho \frac{x}{v}}$$

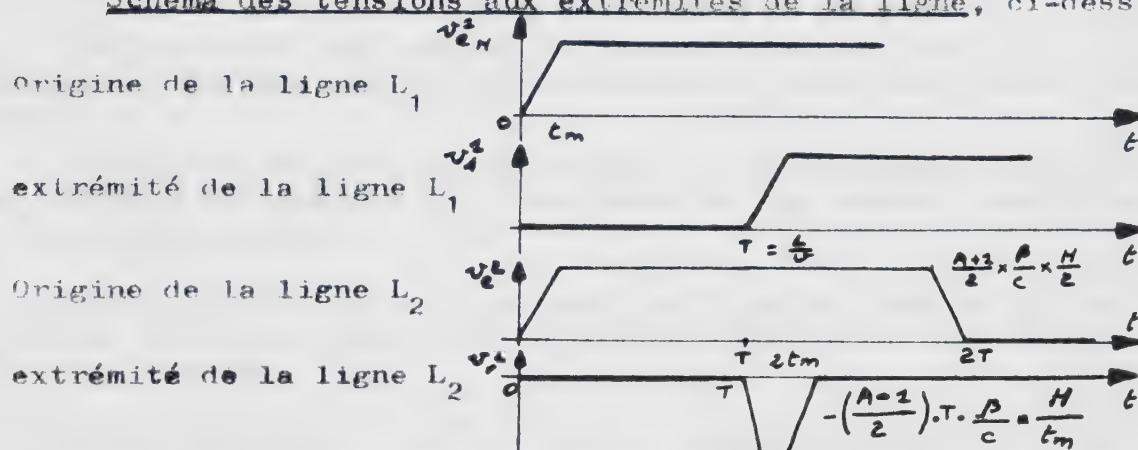
à l'origine de la ligne L_2 ($x = 0$) , l'onde arrière s'écrit :

$$V_e^2 = -\frac{\beta}{c} \times \frac{H}{2} \times \frac{A+1}{2} \left[1 - e^{-\rho 2T} \right]$$

à l'extrémité de la ligne L_2 ($x = L$), l'onde avant s'écrit :

$$V_r^2 = -\frac{\beta}{c} \times T \cdot \frac{A-1}{2} \cdot \rho H \cdot e^{-\rho T}$$

Schéma des tensions aux extrémités de la ligne, ci-dessous



Nous avons supposé que le temps de montée de l'échelon était de l'ordre de $t_m \ll T$ et que le flanc de montée n'était pas déformé par la ligne.

Conclusions :

Dans le cas habituel les lignes L_1 et L_2 sont placées dans un milieu homogène (de l'air), nous pouvons alors poser $A = 1$, l'onde en extrémité de ligne L_2 est nulle.

Si nous remarquons que la ligne L_2 peut être au niveau B ou au niveau H et que le récepteur (porte logique) peut être situé soit du côté de l'origine de L_2 , soit du côté de l'extrémité L_2 , toutes les configurations de tension aux bornes du récepteur sont possibles.

L'on retrouve facilement la configuration des parasites et leurs amplitudes.

L'on remarquera que si l'émetteur est une bascule bistable, le parasite à l'émetteur se boucle dans la bascule et peut conduire à un changement d'état de celle-ci, alors que les entrées ne sont pas parasitées.

Si l'on veut diminuer ces parasites, il est nécessaire de diminuer la mutuelle inductance linéique et de diminuer la capacité de couplage β , la solution la plus simple étant d'utiliser des câbles coaxiaux pour les lignes L_1 et L_2 .

Il est possible de modifier les résistances d'extrémité de ligne mais alors, dans ce cas, la ligne n'est plus adaptée et il est nécessaire de tenir compte des réflexions en extrémités des lignes.

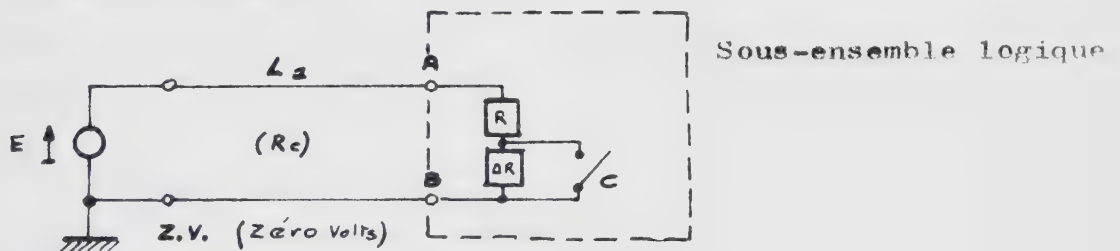
A-3-5- Perturbations dues à l'alimentation :

Il existe deux perturbations essentielles dues au réseau d'alimentation des circuits logiques :

- les perturbations venant du réseau 50 Hz et traversant l'alimentation (macrocoupures, microcoupures et parasites haute fréquence)
- les perturbations dues aux lignes d'alimentation soumise à des variations brusques de l'impédance de la charge (transition des opérateurs logiques).

Lors de la conception d'une machine l'on se protège des perturbations du premier type mais l'on omet souvent de se protéger contre les secondes.

Considérons donc un sous ensemble logique alimenté par l'intermédiaire d'une ligne L_1 . Lors des changements d'états des transistors l'impédance du sous ensemble varie brusquement de $R + \Delta R$ à R , ou réciproquement.



Lors de l'ouverture, ou la fermeture du contact C , les potentiels V_A et V_B varient, d'où une variation des potentiels des signaux émis par le sous ensemble logique vers d'autre sous-ensembles.

D'après la méthode de J. HUG, nous pouvons remplacer la ligne L_1 d'impédance caractéristique R_c de durée de propagation τ par une self équivalente $L = \tau \cdot R_c$ (les impédances du générateur et du récepteur étant faibles devant R_c).

...

La figure 2 représente le schéma équivalent, mais afin de simplifier les calculs nous étudierons la figure 3.

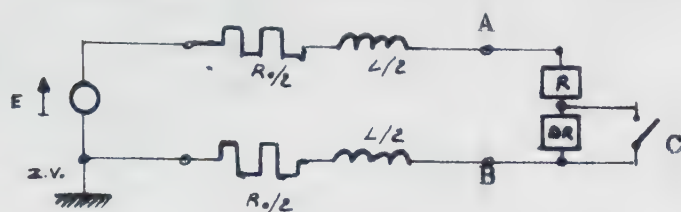


figure 2

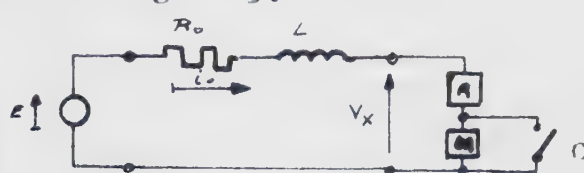


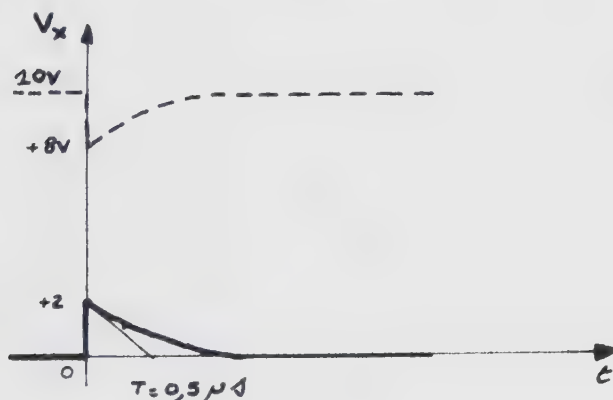
figure 3

Si à l'époque $t = (0)$ l'on ferme C, nous avons :

$$\left[\begin{aligned} V_x &= \frac{R}{R_0 + R} \times \frac{E - Li_0 p}{p \left[1 + \frac{L}{R_0 + R} p \right]} \\ V_x(\infty) &= \frac{E \cdot R}{R_0 + R} \end{aligned} \right.$$

$$\left[\begin{aligned} i_0^- &= i_0^+ = \frac{E}{R_0 + R + \Delta R} \\ V_x(0^-) &= \frac{E(R + \Delta R)}{R_0 + R + \Delta R} \\ V_x(0^+) &= V_x(0^-) \left[1 - \frac{\Delta R}{R_0 + R + \Delta R} \right] \end{aligned} \right.$$

Si l'on remarque que les variations de V_x se partagent entre V_A et V_B , nous avons le graphe suivant pour V_A^x et V_B (représenté pour $R_0 \ll R = 4 \Omega$; $\Delta R = 2 \Omega$; $R_c = 250 \Omega/m$; $\tau \approx 10 \text{ ns}$; $E=10V$)



$$T = \frac{L}{R_0 + R} = \frac{R_c}{R_0 + R} = 500 \text{ ns}$$

D'où tout signal reçu ou émis par ce sous-ensemble sera un signal perturbé par rapport au potentiel de référence de la machine (z.v.)

L'on peut diminuer les perturbations, en ajoutant en parallèle aux bornes du sous-ensemble une capacité dite de stockage C calculée de manière telle que l'amplitude de la variation soit réduite et que le système n'aie pas d'oscillation (2ème ordre amorti à 1 ou 0,7).

Désignation	Schema Electronique	Table des Niveaux	Equation Logique																																				
Porte à diodes dite POSITIVE		<table><tr><th>V_{e_1}</th><th>V_{e_2}</th><th>V_{e_3}</th><th>V_S</th></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	V_{e_1}	V_{e_2}	V_{e_3}	V_S	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0	0	1	0	1	0	1	1	0	0	1	1	1	1	Logique POSITIVE $s = \prod_{i=1}^n e_i$
			V_{e_1}	V_{e_2}	V_{e_3}	V_S																																	
0	0	0	0																																				
0	0	1	0																																				
0	1	0	0																																				
0	1	1	0																																				
1	0	0	0																																				
1	0	1	0																																				
1	1	0	0																																				
1	1	1	1																																				
			Logique NÉGATIVE $s = \sum_{i=1}^n e_i$																																				
Porte à diodes dite NÉGATIVE		<table><tr><th>V_{e_1}</th><th>V_{e_2}</th><th>V_{e_3}</th><th>V_S</th></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	V_{e_1}	V_{e_2}	V_{e_3}	V_S	0	0	0	0	0	0	1	1	0	1	0	1	0	1	1	1	1	0	0	1	1	0	1	1	1	1	0	1	1	1	1	1	Logique POSITIVE $s = \sum_{i=1}^n e_i$
			V_{e_1}	V_{e_2}	V_{e_3}	V_S																																	
0	0	0	0																																				
0	0	1	1																																				
0	1	0	1																																				
0	1	1	1																																				
1	0	0	1																																				
1	0	1	1																																				
1	1	0	1																																				
1	1	1	1																																				
			Logique NÉGATIVE $s = \prod_{i=1}^n e_i$																																				

signation	Schéma Logique	Schéma électronique	Equation (Logique positive)
Inverseur			$S = \bar{A}$
Amplificateur			$S = A$
ET			$s = \prod_{i=1}^{i=30} e_i$
ET Δ			$s = \prod_{i=1}^{i=30} e_i$ (Connecté à la Résistance R_g)
OU			$s = \sum_{i=1}^{i=15} e_i$ (Connecté à la résistance $R_b \dots$)
OU Spécial			$s = \sum_{i=1}^{i=15} e_i$ (Connecté à la Résistance R_b)

Designation	Schéma Logique	Schéma électronique	Equation (Logique positive)
ET ET _Δ			$s = \prod a_i \cdot \prod b_j$
OU OU			$s = \sum a_i + \sum b_j$ (s Connecté à la Résistance R _b)
OU ET _Δ			$s = \sum a_i \cdot \prod b_j$ (s Connecté à la Résistance R _b)
ET OU Spécial			$s = \prod a_i + \prod b_j$ (s Connecté à la Résistance R _b)
ET cable 1/2 OU Fantôme			$s = \prod a_i + \prod b_j$ (s Connecté à la Résistance R _b)
Conditionneur type			$s = \prod a \cdot [\sum b + \prod c + \prod d \cdot \prod e + \prod f \cdot (\prod g + \prod h + \sum k)]$ si S est connecté à la résistance R _b ;
Conditionneurs de base	PLANCHE A-3		

PERMIS D'IMPRIMER

DOCTORAT : DOCTEUR - INGENIEUR

THESE DE : /

NOM : GROSSETETE

Prénoms : Michel

né le 9 AOÛT 1937 à MONTBARD

Dép^t : COTE D'OR

1ère THESE : " CONTRIBUTION A LA SYNTHÈSE DES MACHINES SEQUENTIELLES
APPLIQUEE A LA COMMANDE NUMERIQUE DES MACHINES-OUTILS. "

2e THESE (Propositions faites par l'Université) :



Vu et Approuvé

PARIS, le 9 MARS 1971.

Le Président

de l'Université de PARIS VI

PERMIS D'IMPRIMER

Académie de Paris
UNIVERSITÉ DE PARIS VI
Quai Saint-Benoît - Paris 6
SERVICE DE LA SCOLARITÉ
des Enseignements du 3e Cycle
336 52 52 et 328 12 51

DOCTORAT :	DOCTEUR - INGÉNIEUR
THÈSE DE :	

NOM : GROSSETÊTE
Prénoms : Michel
Né le 2 AGOUT 1937 à MONTARD
Dép. : COTE D'OR

La thèse : "CONTRIBUTION A LA SYNTHESE DES MACHINES SEQUENTIELLES
APPLIQUEE A LA COMMANDE NUMERIQUE DES MACHINES-OUTILS, "

La thèse (proposition faite par l'Université) :

Vu et approuvé
Paris le 9 Mars 1971
Le Président
de l'Université de Paris VI

